# FlashFeed - Master Implementation Guide

**📋** Übersicht

**Dieser Master-Guide führt alle technischen Spezifikationen zusammen und bietet eine Schritt-für-Schritt-Anleitung zur Implementierung des FlashFeed-Prototyps.**

## 📚 Dokumenten-Übersicht

### 1. Requirements & Design

**Requirements-Dokument (Version 2.0): Hierarchische Requirements mit REQ-3.x.x-Nummerierung**

**UI Design Spezifikationen: Farben, Typografie, Komponenten, Responsive Design**

**API Spezifikationen: REST-Endpoints, WebSocket-Events, Error-Handling**

### 2. Daten & Mock-System

**Datenbank Schema: SQLite-basiert für LocalStorage-Integration**

**Mock-Daten Spezifikationen: 500+ Produkte, Timer-basierte Updates, realistische Algorithmen**

### 3. Implementation Details

**Flutter Projektstruktur: Vollständige Ordnerhierarchie und pubspec.yaml**

**BLoC & Widget Implementation: State Management, UI-Komponenten, Event-Handling**

**Repository & Deployment: GitHub Actions, LocalStorage-Integration**

## 🚀 Implementierungsreihenfolge

**Phase 1: Projekt-Setup (Tag 1-2)**

**bash# 1. Flutter-Projekt erstellen**

**flutter create flashfeed**

**cd flashfeed**

## # 2. pubspec.yaml ersetzen (siehe Projektstruktur-Dokument)

## # 3. Ordnerstruktur anlegen

**mkdir -p lib/core/constants lib/core/theme lib/core/services lib/core/utils**

**mkdir -p lib/data/models lib/data/repositories lib/data/datasources**

**mkdir -p lib/presentation/blocs lib/presentation/screens lib/presentation/widgets**

**mkdir -p assets/images/logos assets/data assets/data/floorplans**

## # 4. Dependencies installieren

**flutter pub get**

**flutter pub run build\_runner build --delete-conflicting-outputs**

### Phase 2: Core Foundation (Tag 3-5)

**Reihenfolge: Constants → Models → Services → Storage**

**Constants implementieren: app\_colors.dart, app\_text\_styles.dart, app\_dimensions.dart**

**Models erstellen: Alle @JsonSerializable-Models (Chain, Store, Product, FlashDeal, etc.)**

**Storage Service: LocalStorage-Wrapper für SharedPreferences**

**Mock Data Service: Datengeneration und Timer-Management**

### Phase 3: Data Layer (Tag 6-8)

**Reihenfolge: DataSources → Repositories → Integration Testing**

**LocalStorageDataSource: Vollständige CRUD-Operationen**

**Repositories: Chain-, Store-, FlashDeal-, User-Repository**

**Mock-Daten-Integration: JSON-Assets und Generator-Algorithmen**

### Phase 4: BLoC Layer (Tag 9-12)

**Reihenfolge: Navigation → FlashDeals → Offers → Map**

**MainNavigationBloc: Tab-Switching und Settings-Overlay**

**FlashDealsBloc: Timer-Updates und Countdown-Management**

**OffersBloc: Chain-Selection und Category-Filtering**

**MapBloc: Location Services und Store-Pins**

### Phase 5: UI Implementation (Tag 13-18)

**Reihenfolge: Common Widgets → Screens → Integration**

**Common Widgets: AppHeader, TabNavigation, CountdownTimer**

**Screen Implementation: MainScreen, OffersScreen, MapScreen, FlashDealsScreen**

**Specialized Widgets: FlashDealCard, ChainSelector, StoreMap**

**UI/UX Polish: Animations, Loading States, Error Handling**

### Phase 6: Advanced Features (Tag 19-22)

**Reihenfolge: Maps → Floorplans → Notifications → Optimization**

**Google Maps Integration: Store-Pins, Radius-Filter, Navigation**

**Floorplan System: SVG-Rendering, Beacon-Simulation, Product-Marker**

**Push Notifications: Web-Notifications für Flash-Deals**

**Performance Optimization: Lazy Loading, Caching, State Persistence**

### Phase 7: Deployment (Tag 23-25)

**Reihenfolge: Testing → Build → GitHub Setup → QR Generation**

**Testing: Unit Tests, Widget Tests, Integration Tests**

**Web Build Configuration: base-href, Service Worker, PWA-Manifest**

**GitHub Actions Setup: Automated Deployment Pipeline**

**QR Code Generation: Mobile-Demo-Zugang**

## 💻 Entwicklungsumgebung Setup

**Voraussetzungen**

**bash# Flutter SDK (3.16.0+)**

**flutter --version**

**# Git**

**git --version**

**# Python (für QR-Code-Generation)**

**python3 --version**

**pip install qrcode[pil]**

**# Code Editor: VS Code mit Flutter Extension**

**IDE-Konfiguration (VS Code)**

**.vscode/settings.json:**

**json{**

**"editor.codeActionsOnSave": {**

**"source.fixAll": true**

**},**

**"dart.previewFlutterUiGuides": true,**

**"dart.previewFlutterUiGuidesCustomTracking": true,**

**"editor.rulers": [80, 120],**

**"dart.lineLength": 120**

**}**

**.vscode/launch.json:**

**json{**

**"version": "0.2.0",**

**"configurations": [**

**{**

**"name": "FlashFeed Web",**

**"request": "launch",**

**"type": "dart",**

**"args": ["--web-port", "3000"],**

**"deviceId": "chrome"**

**}**

**]**

**}**

**🧪 Testing-Strategie**

**Unit Tests**

**dart// test/models/flash\_deal\_test.dart**

**void main() {**

**group('FlashDeal', () {**

**test('should create valid FlashDeal from JSON', () {**

**final json = {...}; // Mock JSON**

**final flashDeal = FlashDeal.fromJson(json);**

**expect(flashDeal.id, 'flash\_001');**

**expect(flashDeal.discountPercentage, 50);**

**});**

**test('should calculate remaining seconds correctly', () {**

**final futureDate = DateTime.now().add(Duration(hours: 2));**

**final flashDeal = FlashDeal(expiresAt: futureDate, ...);**

**expect(flashDeal.remainingSeconds, greaterThan(7000));**

**});**

**});**

**}**

**Widget Tests**

**dart// test/widgets/flash\_deal\_card\_test.dart**

**void main() {**

**testWidgets('FlashDealCard displays countdown timer', (tester) async {**

**final flashDeal = FlashDeal(...);**

**await tester.pumpWidget(**

**MaterialApp(**

**home: FlashDealCard(flashDeal: flashDeal),**

**),**

**);**

**expect(find.byType(CountdownTimer), findsOneWidget);**

**expect(find.text('-50%'), findsOneWidget);**

**});**

**}**

**Integration Tests**

**dart// integration\_test/app\_test.dart**

**void main() {**

**IntegrationTestWidgetsFlutterBinding.ensureInitialized();**

**testWidgets('Full user flow: Tab navigation and flash deal interaction', (tester) async {**

**app.main();**

**await tester.pumpAndSettle();**

**// Test tab navigation**

**expect(find.text('Angebote'), findsOneWidget);**

**await tester.tap(find.text('Flash'));**

**await tester.pumpAndSettle();**

**// Test flash deal interaction**

**expect(find.byType(FlashDealCard), findsWidgets);**

**await tester.tap(find.text('Zum Lageplan').first);**

**await tester.pumpAndSettle();**

**expect(find.byType(FloorplanWidget), findsOneWidget);**

**});**

**}**

**⚡ Performance-Optimierung**

**1. Lazy Loading**

**dartclass OffersScreen extends StatelessWidget {**

**@override**

**Widget build(BuildContext context) {**

**return BlocBuilder<OffersBloc, OffersState>(**

**builder: (context, state) {**

**if (state is OffersLoading) {**

**return SkeletonLoader(); // Statt CircularProgressIndicator**

**}**

**return LazyLoadListView(**

**itemCount: state.offers.length,**

**itemBuilder: (context, index) => OfferCard(state.offers[index]),**

**);**

**},**

**);**

**}**

**}**

**2. State Persistence**

**dartclass FlashDealsBlocObserver extends BlocObserver {**

**@override**

**void onChange(BlocBase bloc, Change change) {**

**if (bloc is FlashDealsBloc && change.nextState is FlashDealsLoaded) {**

**// Persist state to LocalStorage**

**StorageService.instance.storeJson('flash\_deals\_state',**

**(change.nextState as FlashDealsLoaded).toJson());**

**}**

**super.onChange(bloc, change);**

**}**

**}**

**3. Image Optimization**

**dartWidget chainIcon(Chain chain) {**

**return CachedNetworkImage(**

**imageUrl: chain.logoUrl,**

**width: 60,**

**height: 60,**

**fit: BoxFit.cover,**

**placeholder: (context, url) => SkeletonContainer(60, 60),**

**errorWidget: (context, url, error) => DefaultChainIcon(),**

**memCacheWidth: 120, // 2x für Retina**

**memCacheHeight: 120,**

**);**

**}**

**🐛 Debugging & Troubleshooting**

**Häufige Probleme**

**1. Mock-Daten werden nicht geladen**

**dart// Debug: Check LocalStorage content**

**final storage = StorageService.instance;**

**final chains = storage.getList('flashfeed\_chains', (json) => Chain.fromJson(json));**

**print('Loaded ${chains.length} chains from storage');**

**2. Timer-Updates funktionieren nicht**

**dart// Debug: Verify timer state**

**class FlashDealsBloc extends Bloc<FlashDealsEvent, FlashDealsState> {**

**@override**

**void add(FlashDealsEvent event) {**

**print('FlashDealsBloc: Adding event $event');**

**super.add(event);**

**}**

**}**

**3. GitHub Pages Deployment-Fehler**

**yaml# .github/workflows/deploy.yml - Add debugging**

**- name: Debug Build Output**

**run: |**

**ls -la build/web/**

**cat build/web/index.html | head -20**

**Debug-Tools Setup**

**dartvoid main() {**

**if (kDebugMode) {**

**// Bloc Debugging**

**Bloc.observer = AppBlocObserver();**

**// HTTP Logging**

**HttpOverrides.global = LoggingHttpOverrides();**

**}**

**runApp(FlashFeedApp());**

**}**

**📱 Mobile Optimization**

**Responsive Breakpoints**

**dartclass ResponsiveBreakpoints {**

**static const double mobile = 768;**

**static const double tablet = 1024;**

**static const double desktop = 1440;**

**static bool isMobile(BuildContext context) {**

**return MediaQuery.of(context).size.width < mobile;**

**}**

**static bool isTablet(BuildContext context) {**

**final width = MediaQuery.of(context).size.width;**

**return width >= mobile && width < desktop;**

**}**

**}**

**Touch Optimizations**

**dartWidget touchOptimizedButton(String text, VoidCallback onPressed) {**

**return Container(**

**constraints: BoxConstraints(minHeight: 44, minWidth: 44), // A11y**

**child: ElevatedButton(**

**onPressed: onPressed,**

**child: Text(text),**

**style: ElevatedButton.styleFrom(**

**tapTargetSize: MaterialTapTargetSize.expandedTouchTarget,**

**),**

**),**

**);**

**}**

**🚀 Deployment-Checkliste**

**Pre-Deployment**

**Alle Tests bestehen (flutter test)**

**Build erfolgreich (flutter build web)**

**Assets sind verfügbar (Händler-Logos, Icons)**

**Google Maps API Key konfiguriert**

**Base-href für Repository gesetzt**

**Service Worker aktiviert**

**GitHub Setup**

**Repository ist public**

**GitHub Actions aktiviert**

**Pages-Settings konfiguriert (Source: GitHub Actions)**

**Secrets konfiguriert (falls nötig)**

**Post-Deployment**

**URL funktioniert (https://username.github.io/flashfeed-prototype/)**

**Mobile Responsiveness testen**

**PWA-Installation testen**

**QR-Code generiert und getestet**

**Performance mit Lighthouse testen (Score > 90)**

**Demo-Vorbereitung**

**Mock-Daten sind realistisch und vielfältig**

**Flash-Deal-Timer laufen korrekt**

**Alle drei Panels funktionieren**

**Beacon-Simulation funktioniert ("Filiale betreten"-Button)**

**Karten-Integration zeigt Stores korrekt**

**Einkaufslisten-Demo ist vorbereitet**

**📊 Erfolgs-Metriken**

**Technische KPIs**

**Build-Zeit: < 3 Minuten in GitHub Actions**

**App-Startzeit: < 3 Sekunden auf 3G-Verbindung**

**Bundle-Größe: < 2MB für Initial Load**

**Lighthouse Score: Performance > 90, Accessibility > 95**

**Demo-Metriken**

**Feature-Abdeckung: Alle 70+ Requirements funktional demonstriert**

**Mock-Daten-Qualität: Realistisch genug für Geschäftsmodell-Validierung**

**User Journey: Komplette Flows in < 30 Sekunden navigierbar**

**Cross-Device: Funktioniert auf Desktop, Tablet, Mobile**

# 🎯 Success Criteria

**Ein erfolgreich implementierter FlashFeed-Prototyp erfüllt:**

**Funktionale Criteria**

**Alle drei Panels sind vollständig navigierbar**

**Flash-Deals updaten automatisch mit Countdown**

**Karte zeigt realistische Filialstandorte**

**Beacon-Simulation aktiviert Lageplan-Ansicht**

**Responsive Design funktioniert auf allen Geräten**

**Technische Criteria**

**GitHub Pages Deployment läuft automatisch**

**QR-Code ermöglicht sofortigen Mobile-Zugang**

**Alle BLoCs arbeiten korrekt mit LocalStorage**

**Mock-Timer-System aktualisiert Daten zuverlässig**

**PWA-Features sind aktiviert (Offline, Install)**

**Präsentations-Criteria**

**Professor kann alle Features in 5 Minuten testen**

**Geschäftsmodell-Konzepte sind klar demonstriert**

**Two-sided Market (B2B/B2C) ist erkennbar**

**Nachhaltigkeits-Aspekt (Food Waste) ist sichtbar**

**Technische Kompetenz wird überzeugend gezeigt**

**🎉 Abschluss**

**Mit diesen 7 technischen Spezifikations-Dokumenten haben Sie alles Notwendige für eine vollständige Implementation des FlashFeed-Prototyps:**

**Requirements-Dokument - Was gebaut werden soll**

**UI Design Spezifikationen - Wie es aussehen soll**

**API Spezifikationen - Wie Daten fließen**

**Datenbank Schema - Wie Daten strukturiert sind**

**Mock-Daten Spezifikationen - Womit getestet wird**

**Flutter Implementierung - Wie es gebaut wird**

**Repository & Deployment - Wie es deployed wird**

**Geschätzte Gesamtentwicklungszeit: 20-25 Arbeitstage für einen erfahrenen Flutter-Entwickler**

**Nächste Schritte:**

**Repository auf GitHub erstellen**

**Flutter-Projekt nach Spezifikation aufsetzen**

**Phase für Phase implementieren**

**Kontinuierlich testen und deploywen**

**QR-Code für Professor-Demo generieren**

**Ihr FlashFeed-Prototyp wird eine überzeugende Demonstration Ihres digitalen Geschäftsmodells und Ihrer technischen Umsetzungskompetenz sein.**

**Master Guide Version: 1.0**

**Vollständigkeit: Implementierungsbereit**

**Geschätzte Umsetzungszeit: 20-25 Tage**

**Nächster Schritt: Flutter-Projekt Setup beginnen**

# Requirements & Design

Siehe FlashFeed - Master Implementation Guide

📋 Übersicht

Dieser Master-Guide führt alle technischen Spezifikationen zusammen und bietet eine Schritt-für-Schritt-Anleitung zur Implementierung des FlashFeed-Prototyps.

📚 Dokumenten-Übersicht

1. Requirements & Design

Requirements-Dokument (Version 2.0): Hierarchische Requirements mit REQ-3.x.x-Nummerierung

UI Design Spezifikationen: Farben, Typografie, Komponenten, Responsive Design

API Spezifikationen: REST-Endpoints, WebSocket-Events, Error-Handling

2. Daten & Mock-System

Datenbank Schema: SQLite-basiert für LocalStorage-Integration

Mock-Daten Spezifikationen: 500+ Produkte, Timer-basierte Updates, realistische Algorithmen

3. Implementation Details

Flutter Projektstruktur: Vollständige Ordnerhierarchie und pubspec.yaml

BLoC & Widget Implementation: State Management, UI-Komponenten, Event-Handling

Repository & Deployment: GitHub Actions, LocalStorage-Integration

🚀 Implementierungsreihenfolge

Phase 1: Projekt-Setup (Tag 1-2)

bash# 1. Flutter-Projekt erstellen

flutter create flashfeed\_prototype

cd flashfeed\_prototype

# 2. pubspec.yaml ersetzen (siehe Projektstruktur-Dokument)

# 3. Ordnerstruktur anlegen

mkdir -p lib/core/constants lib/core/theme lib/core/services lib/core/utils

mkdir -p lib/data/models lib/data/repositories lib/data/datasources

mkdir -p lib/presentation/blocs lib/presentation/screens lib/presentation/widgets

mkdir -p assets/images/logos assets/data assets/data/floorplans

# 4. Dependencies installieren

flutter pub get

flutter pub run build\_runner build --delete-conflicting-outputs

Phase 2: Core Foundation (Tag 3-5)

Reihenfolge: Constants → Models → Services → Storage

Constants implementieren: app\_colors.dart, app\_text\_styles.dart, app\_dimensions.dart

Models erstellen: Alle @JsonSerializable-Models (Chain, Store, Product, FlashDeal, etc.)

Storage Service: LocalStorage-Wrapper für SharedPreferences

Mock Data Service: Datengeneration und Timer-Management

Phase 3: Data Layer (Tag 6-8)

Reihenfolge: DataSources → Repositories → Integration Testing

LocalStorageDataSource: Vollständige CRUD-Operationen

Repositories: Chain-, Store-, FlashDeal-, User-Repository

Mock-Daten-Integration: JSON-Assets und Generator-Algorithmen

Phase 4: BLoC Layer (Tag 9-12)

Reihenfolge: Navigation → FlashDeals → Offers → Map

MainNavigationBloc: Tab-Switching und Settings-Overlay

FlashDealsBloc: Timer-Updates und Countdown-Management

OffersBloc: Chain-Selection und Category-Filtering

MapBloc: Location Services und Store-Pins

Phase 5: UI Implementation (Tag 13-18)

Reihenfolge: Common Widgets → Screens → Integration

Common Widgets: AppHeader, TabNavigation, CountdownTimer

Screen Implementation: MainScreen, OffersScreen, MapScreen, FlashDealsScreen

Specialized Widgets: FlashDealCard, ChainSelector, StoreMap

UI/UX Polish: Animations, Loading States, Error Handling

Phase 6: Advanced Features (Tag 19-22)

Reihenfolge: Maps → Floorplans → Notifications → Optimization

Google Maps Integration: Store-Pins, Radius-Filter, Navigation

Floorplan System: SVG-Rendering, Beacon-Simulation, Product-Marker

Push Notifications: Web-Notifications für Flash-Deals

Performance Optimization: Lazy Loading, Caching, State Persistence

Phase 7: Deployment (Tag 23-25)

Reihenfolge: Testing → Build → GitHub Setup → QR Generation

Testing: Unit Tests, Widget Tests, Integration Tests

Web Build Configuration: base-href, Service Worker, PWA-Manifest

GitHub Actions Setup: Automated Deployment Pipeline

QR Code Generation: Mobile-Demo-Zugang

💻 Entwicklungsumgebung Setup

Voraussetzungen

bash# Flutter SDK (3.16.0+)

flutter --version

# Git

git --version

# Python (für QR-Code-Generation)

python3 --version

pip install qrcode[pil]

# Code Editor: VS Code mit Flutter Extension

IDE-Konfiguration (VS Code)

.vscode/settings.json:

json{

"editor.codeActionsOnSave": {

"source.fixAll": true

},

"dart.previewFlutterUiGuides": true,

"dart.previewFlutterUiGuidesCustomTracking": true,

"editor.rulers": [80, 120],

"dart.lineLength": 120

}

.vscode/launch.json:

json{

"version": "0.2.0",

"configurations": [

{

"name": "FlashFeed Web",

"request": "launch",

"type": "dart",

"args": ["--web-port", "3000"],

"deviceId": "chrome"

}

]

}

🧪 Testing-Strategie

Unit Tests

dart// test/models/flash\_deal\_test.dart

void main() {

group('FlashDeal', () {

test('should create valid FlashDeal from JSON', () {

final json = {...}; // Mock JSON

final flashDeal = FlashDeal.fromJson(json);

expect(flashDeal.id, 'flash\_001');

expect(flashDeal.discountPercentage, 50);

});

test('should calculate remaining seconds correctly', () {

final futureDate = DateTime.now().add(Duration(hours: 2));

final flashDeal = FlashDeal(expiresAt: futureDate, ...);

expect(flashDeal.remainingSeconds, greaterThan(7000));

});

});

}

Widget Tests

dart// test/widgets/flash\_deal\_card\_test.dart

void main() {

testWidgets('FlashDealCard displays countdown timer', (tester) async {

final flashDeal = FlashDeal(...);

await tester.pumpWidget(

MaterialApp(

home: FlashDealCard(flashDeal: flashDeal),

),

);

expect(find.byType(CountdownTimer), findsOneWidget);

expect(find.text('-50%'), findsOneWidget);

});

}

Integration Tests

dart// integration\_test/app\_test.dart

void main() {

IntegrationTestWidgetsFlutterBinding.ensureInitialized();

testWidgets('Full user flow: Tab navigation and flash deal interaction', (tester) async {

app.main();

await tester.pumpAndSettle();

// Test tab navigation

expect(find.text('Angebote'), findsOneWidget);

await tester.tap(find.text('Flash'));

await tester.pumpAndSettle();

// Test flash deal interaction

expect(find.byType(FlashDealCard), findsWidgets);

await tester.tap(find.text('Zum Lageplan').first);

await tester.pumpAndSettle();

expect(find.byType(FloorplanWidget), findsOneWidget);

});

}

⚡ Performance-Optimierung

1. Lazy Loading

dartclass OffersScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return BlocBuilder<OffersBloc, OffersState>(

builder: (context, state) {

if (state is OffersLoading) {

return SkeletonLoader(); // Statt CircularProgressIndicator

}

return LazyLoadListView(

itemCount: state.offers.length,

itemBuilder: (context, index) => OfferCard(state.offers[index]),

);

},

);

}

}

2. State Persistence

dartclass FlashDealsBlocObserver extends BlocObserver {

@override

void onChange(BlocBase bloc, Change change) {

if (bloc is FlashDealsBloc && change.nextState is FlashDealsLoaded) {

// Persist state to LocalStorage

StorageService.instance.storeJson('flash\_deals\_state',

(change.nextState as FlashDealsLoaded).toJson());

}

super.onChange(bloc, change);

}

}

3. Image Optimization

dartWidget chainIcon(Chain chain) {

return CachedNetworkImage(

imageUrl: chain.logoUrl,

width: 60,

height: 60,

fit: BoxFit.cover,

placeholder: (context, url) => SkeletonContainer(60, 60),

errorWidget: (context, url, error) => DefaultChainIcon(),

memCacheWidth: 120, // 2x für Retina

memCacheHeight: 120,

);

}

🐛 Debugging & Troubleshooting

Häufige Probleme

1. Mock-Daten werden nicht geladen

dart// Debug: Check LocalStorage content

final storage = StorageService.instance;

final chains = storage.getList('flashfeed\_chains', (json) => Chain.fromJson(json));

print('Loaded ${chains.length} chains from storage');

2. Timer-Updates funktionieren nicht

dart// Debug: Verify timer state

class FlashDealsBloc extends Bloc<FlashDealsEvent, FlashDealsState> {

@override

void add(FlashDealsEvent event) {

print('FlashDealsBloc: Adding event $event');

super.add(event);

}

}

3. GitHub Pages Deployment-Fehler

yaml# .github/workflows/deploy.yml - Add debugging

- name: Debug Build Output

run: |

ls -la build/web/

cat build/web/index.html | head -20

Debug-Tools Setup

dartvoid main() {

if (kDebugMode) {

// Bloc Debugging

Bloc.observer = AppBlocObserver();

// HTTP Logging

HttpOverrides.global = LoggingHttpOverrides();

}

runApp(FlashFeedApp());

}

📱 Mobile Optimization

Responsive Breakpoints

dartclass ResponsiveBreakpoints {

static const double mobile = 768;

static const double tablet = 1024;

static const double desktop = 1440;

static bool isMobile(BuildContext context) {

return MediaQuery.of(context).size.width < mobile;

}

static bool isTablet(BuildContext context) {

final width = MediaQuery.of(context).size.width;

return width >= mobile && width < desktop;

}

}

Touch Optimizations

dartWidget touchOptimizedButton(String text, VoidCallback onPressed) {

return Container(

constraints: BoxConstraints(minHeight: 44, minWidth: 44), // A11y

child: ElevatedButton(

onPressed: onPressed,

child: Text(text),

style: ElevatedButton.styleFrom(

tapTargetSize: MaterialTapTargetSize.expandedTouchTarget,

),

),

);

}

🚀 Deployment-Checkliste

Pre-Deployment

Alle Tests bestehen (flutter test)

Build erfolgreich (flutter build web)

Assets sind verfügbar (Händler-Logos, Icons)

Google Maps API Key konfiguriert

Base-href für Repository gesetzt

Service Worker aktiviert

GitHub Setup

Repository ist public

GitHub Actions aktiviert

Pages-Settings konfiguriert (Source: GitHub Actions)

Secrets konfiguriert (falls nötig)

Post-Deployment

URL funktioniert (https://username.github.io/flashfeed-prototype/)

Mobile Responsiveness testen

PWA-Installation testen

QR-Code generiert und getestet

Performance mit Lighthouse testen (Score > 90)

Demo-Vorbereitung

Mock-Daten sind realistisch und vielfältig

Flash-Deal-Timer laufen korrekt

Alle drei Panels funktionieren

Beacon-Simulation funktioniert ("Filiale betreten"-Button)

Karten-Integration zeigt Stores korrekt

Einkaufslisten-Demo ist vorbereitet

📊 Erfolgs-Metriken

Technische KPIs

Build-Zeit: < 3 Minuten in GitHub Actions

App-Startzeit: < 3 Sekunden auf 3G-Verbindung

Bundle-Größe: < 2MB für Initial Load

Lighthouse Score: Performance > 90, Accessibility > 95

Demo-Metriken

Feature-Abdeckung: Alle 70+ Requirements funktional demonstriert

Mock-Daten-Qualität: Realistisch genug für Geschäftsmodell-Validierung

User Journey: Komplette Flows in < 30 Sekunden navigierbar

Cross-Device: Funktioniert auf Desktop, Tablet, Mobile

🎯 Success Criteria

Ein erfolgreich implementierter FlashFeed-Prototyp erfüllt:

Funktionale Criteria

Alle drei Panels sind vollständig navigierbar

Flash-Deals updaten automatisch mit Countdown

Karte zeigt realistische Filialstandorte

Beacon-Simulation aktiviert Lageplan-Ansicht

Responsive Design funktioniert auf allen Geräten

Technische Criteria

GitHub Pages Deployment läuft automatisch

QR-Code ermöglicht sofortigen Mobile-Zugang

Alle BLoCs arbeiten korrekt mit LocalStorage

Mock-Timer-System aktualisiert Daten zuverlässig

PWA-Features sind aktiviert (Offline, Install)

Präsentations-Criteria

Professor kann alle Features in 5 Minuten testen

Geschäftsmodell-Konzepte sind klar demonstriert

Two-sided Market (B2B/B2C) ist erkennbar

Nachhaltigkeits-Aspekt (Food Waste) ist sichtbar

Technische Kompetenz wird überzeugend gezeigt

🎉 Abschluss

Mit diesen 7 technischen Spezifikations-Dokumenten haben Sie alles Notwendige für eine vollständige Implementation des FlashFeed-Prototyps:

Requirements-Dokument - Was gebaut werden soll

UI Design Spezifikationen - Wie es aussehen soll

API Spezifikationen - Wie Daten fließen

Datenbank Schema - Wie Daten strukturiert sind

Mock-Daten Spezifikationen - Womit getestet wird

Flutter Implementierung - Wie es gebaut wird

Repository & Deployment - Wie es deployed wird

Geschätzte Gesamtentwicklungszeit: 20-25 Arbeitstage für einen erfahrenen Flutter-Entwickler

Nächste Schritte:

Repository auf GitHub erstellen

Flutter-Projekt nach Spezifikation aufsetzen

Phase für Phase implementieren

Kontinuierlich testen und deploywen

QR-Code für Professor-Demo generieren

Ihr FlashFeed-Prototyp wird eine überzeugende Demonstration Ihres digitalen Geschäftsmodells und Ihrer technischen Umsetzungskompetenz sein.

## Requirements

Siehe FlashFeed Requirements.docx

## UI Design Spezifikationen

**1. Design System**

**1.1 Farbschema**

/\* Primary Colors \*/

--primary-green: #2E8B57 /\* SeaGreen - Nachhaltigkeit \*/

--primary-red: #DC143C /\* Crimson - Flash-Rabatte \*/

--primary-blue: #1E90FF /\* DodgerBlue - Vertrauen \*/

/\* Secondary Colors \*/

--secondary-light-green: #90EE90 /\* LightGreen \*/

--secondary-orange: #FF6347 /\* Tomato - Urgency \*/

--secondary-gray: #708090 /\* SlateGray \*/

/\* Neutral Colors \*/

--background-light: #FAFAFA

--background-dark: #2C2C2C

--text-primary: #333333

--text-secondary: #666666

--text-white: #FFFFFF

/\* Status Colors \*/

--success: #28A745

--warning: #FFC107

--error: #DC3545

--info: #17A2B8

**1.2 Typography**

/\* Font Families \*/

--font-primary: 'Roboto', sans-serif

--font-secondary: 'Open Sans', sans-serif

--font-mono: 'Roboto Mono', monospace

/\* Font Sizes \*/

--text-xs: 12px

--text-sm: 14px

--text-base: 16px

--text-lg: 18px

--text-xl: 20px

--text-2xl: 24px

--text-3xl: 30px

--text-4xl: 36px

/\* Font Weights \*/

--weight-light: 300

--weight-normal: 400

--weight-medium: 500

--weight-semibold: 600

--weight-bold: 700

**1.3 Spacing System**

--space-1: 4px

--space-2: 8px

--space-3: 12px

--space-4: 16px

--space-5: 20px

--space-6: 24px

--space-8: 32px

--space-10: 40px

--space-12: 48px

--space-16: 64px

**2. UI-Komponenten-Spezifikationen**

**2.1 Header-Panel (Statisch)**

Höhe: 64px

Background: primary-green

Layout: Flexbox (space-between)

Links:

- App-Logo (32x32px, weiß)

- App-Name "FlashFeed" (text-xl, weight-bold, text-white)

Rechts:

- Hamburger-Menu-Icon (24x24px, text-white)

- Tap-Area: 44x44px (Accessibility)

**2.2 Tab-Navigation (Drei Panels)**

Höhe: 56px

Background: background-light

Border-Top: 1px solid secondary-gray

Tabs (Equal Width):

1. "Angebote" - Icon: shopping-cart (24px)

2. "Karte" - Icon: map-pin (24px)

3. "Flash" - Icon: zap (24px)

Active State:

- Background: primary-green

- Text: text-white

- Border-Bottom: 3px solid primary-red

Inactive State:

- Background: transparent

- Text: text-secondary

- Icon: secondary-gray

**2.3 Panel 1: Angebots-Übersicht**

**2.3.1 Händler-Icon-Leiste**

Höhe: 80px

Background: background-light

Padding: space-3

Layout: Horizontal Scroll

Icon-Container:

- Größe: 60x60px

- Border-Radius: 8px

- Margin-Right: space-3

Active Icon:

- Border: 2px solid primary-green

- Opacity: 1.0

- Drop-Shadow: 0 2px 8px rgba(46,139,87,0.3)

Inactive Icon:

- Border: 1px solid secondary-gray

- Opacity: 0.5

- Grayscale: 100%

**2.3.2 Produktgruppen-Grid**

Layout: CSS Grid

Grid-Template-Columns: repeat(auto-fit, minmax(150px, 1fr))

Gap: space-4

Padding: space-4

Produktgruppen-Karte:

- Min-Height: 120px

- Background: white

- Border-Radius: 12px

- Box-Shadow: 0 2px 8px rgba(0,0,0,0.1)

- Padding: space-4

Header:

- Kategorie-Icon (32x32px, primary-green)

- Kategorie-Name (text-lg, weight-medium)

Content:

- Anzahl Angebote (text-sm, text-secondary)

- Bester Rabatt-Badge (background: secondary-orange, text-white)

**2.4 Panel 2: Karten-Ansicht**

**2.4.1 Map-Container**

Height: calc(100vh - 120px) /\* Full Screen minus Header + Tabs \*/

Background: background-light

Google Maps Integration:

- Default Zoom: 13

- Center: User GPS Position

- Style: Standard (nicht Satellite)

Controls:

- Zoom Controls: Bottom-Right

- Current Location Button: Bottom-Right, above Zoom

**2.4.2 Radius-Filter**

Position: Top-Left Overlay

Background: white

Border-Radius: 8px

Box-Shadow: 0 4px 12px rgba(0,0,0,0.15)

Padding: space-3

Slider:

- Width: 200px

- Track-Color: secondary-gray

- Thumb-Color: primary-blue

- Active-Track-Color: primary-green

Labels:

- "1km" - "20km" (text-sm)

- Current Value (text-base, weight-medium)

**2.4.3 Store-Pins**

Size: 40x40px

Border-Radius: 50% 50% 50% 0% (Teardrop)

Border: 2px solid white

Drop-Shadow: 0 2px 6px rgba(0,0,0,0.3)

Pin Colors by Chain:

- EDEKA: #005CA9 (Blau)

- REWE: #CC071E (Rot)

- ALDI: #00549F (Dunkelblau)

- LIDL: #0050AA (Blau)

- NETTO: #FFD100 (Gelb)

Hover State:

- Scale: 1.2

- Z-Index: 1000

- Animation: bounce 0.3s ease

**2.5 Panel 3: Echtzeit-Rabatte**

**2.5.1 Flash-Rabatt-Karte**

Background: white

Border-Radius: 12px

Margin-Bottom: space-4

Box-Shadow: 0 3px 10px rgba(0,0,0,0.1)

Border-Left: 4px solid primary-red

Layout:

- Padding: space-4

- Display: Flex Column

Header:

- Countdown-Timer (text-2xl, weight-bold, primary-red)

- Rabatt-Badge (background: primary-red, text-white, text-xl)

Content:

- Produktname (text-lg, weight-medium)

- Preis-Vergleich: Durchgestrichen + Neu (text-base)

- Store-Info (text-sm, text-secondary)

Action:

- "Zum Lageplan" Button (primary-green, full-width)

**2.5.2 Countdown-Timer**

Format: "02:34:15" (HH:MM:SS)

Color Coding:

- > 1 Stunde: primary-green

- 30-60 Min: secondary-orange

- < 30 Min: primary-red (blinkend)

Animation:

- Update: every 1 second

- Blink: 0.5s interval when < 30min

**2.5.3 Lageplan-Modal**

Background: rgba(0,0,0,0.8) (Overlay)

Modal-Container:

- Background: white

- Width: 90vw

- Max-Height: 80vh

- Border-Radius: 16px

- Position: Center Screen

Header:

- Filialname (text-xl, weight-bold)

- Close-Button (X, top-right, 32x32px)

Content:

- SVG-Lageplan (responsive)

- Produkt-Marker (pulsing red dot, 16x16px)

- Legende (Regal-Nummern)

Footer:

- "Navigation starten" Button (primary-blue, full-width)

**3. Icon-Spezifikationen**

**3.1 Verwendete Icon-Bibliothek**

 **Lucide React Icons** (24px Standard-Größe)

 Fallback: Material Design Icons

**3.2 Icon-Mapping**

- shopping-cart: Panel 1 (Angebote)

- map-pin: Panel 2 (Karte)

- zap: Panel 3 (Flash-Rabatte)

- menu: Hamburger-Menu

- x: Close-Button

- clock: Countdown-Timer

- navigation: GPS/Navigation

- bell: Push-Notifications

- user: Benutzerprofil

- settings: Einstellungen

- heart: Favoriten

- star: Bewertungen

- share-2: Social Sharing

**3.3 Händler-Logos (60x60px)**

Benötigte Assets:

- edeka-logo.png (60x60, transparent background)

- rewe-logo.png

- aldi-logo.png

- lidl-logo.png

- netto-schwarz-logo.png

- kaufland-logo.png

- penny-logo.png

- real-logo.png

- norma-logo.png

- netto-rot-logo.png

Format: PNG mit transparentem Hintergrund

Auflösung: 120x120px (@2x für Retina)

**4. Responsive Breakpoints**

/\* Mobile First Approach \*/

--mobile: 320px - 768px

--tablet: 768px - 1024px

--desktop: 1024px+

/\* Component Adjustments \*/

@media (max-width: 768px) {

.produktgruppen-grid {

grid-template-columns: repeat(2, 1fr);

}

.händler-icons {

overflow-x: scroll;

}

}

@media (min-width: 1024px) {

.drei-panel-layout {

display: grid;

grid-template-columns: 1fr 2fr 1fr;

}

}

**5. Animation & Transitions**

**5.1 Standard-Übergänge**

--transition-fast: 0.15s ease

--transition-normal: 0.3s ease

--transition-slow: 0.5s ease

/\* Component Transitions \*/

.panel-switch {

transition: transform var(--transition-normal);

}

.flash-rabatt-card {

transition: box-shadow var(--transition-fast);

}

.flash-rabatt-card:hover {

box-shadow: 0 6px 20px rgba(0,0,0,0.15);

}

**5.2 Loading States**

.skeleton-loader {

background: linear-gradient(90deg, #f0f0f0 25%, #e0e0e0 50%, #f0f0f0 75%);

background-size: 200% 100%;

animation: loading 1.5s infinite;

}

@keyframes loading {

0% { background-position: 200% 0; }

100% { background-position: -200% 0; }

}

**6. Accessibility (A11y)**

**6.1 WCAG 2.1 AA Compliance**

- Farbkontrast: Minimum 4.5:1 für normalen Text

- Farbkontrast: Minimum 3:1 für große Texte (>18px)

- Touch-Target: Minimum 44x44px

- Focus-Indicators: 2px solid primary-blue outline

- Alternative Texte für alle Icons

**6.2 Screen Reader Support**

<!-- Beispiel: Countdown-Timer -->

<div aria-live="polite" aria-atomic="true">

<span aria-label="Angebot läuft ab in 2 Stunden 34 Minuten">

02:34:15

</span>

</div>

<!-- Beispiel: Händler-Icon -->

<button aria-label="EDEKA Angebote anzeigen" role="button">

<img src="edeka-logo.png" alt="EDEKA Logo" />

</button>

**7. Error States & Feedback**

**7.1 Error Messages**

.error-message {

background: #FFF5F5;

border: 1px solid #FEB2B2;

color: #C53030;

border-radius: 6px;

padding: space-3;

font-size: var(--text-sm);

}

**7.2 Success Messages**

.success-message {

background: #F0FFF4;

border: 1px solid #9AE6B4;

color: #2F855A;

border-radius: 6px;

padding: space-3;

font-size: var(--text-sm);

}

**7.3 Loading Spinners**

.loading-spinner {

width: 32px;

height: 32px;

border: 3px solid #f3f3f3;

border-top: 3px solid var(--primary-green);

border-radius: 50%;

animation: spin 1s linear infinite;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

### API-Architektur Übersicht

**1.1 Base URL**

Development: http://localhost:3000/api/v1

Production: https://api.flashfeed.de/v1

**1.2 Authentication**

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json

Accept: application/json

**1.3 Response Format**

{

"success": true|false,

"data": {...},

"message": "Success/Error message",

"timestamp": "2025-08-24T10:30:00Z",

"errors": [...] // Optional array for validation errors

}

**2. Core Endpoints**

**2.1 Authentication Endpoints**

**POST /auth/register**

Request:

{

"email": "user@example.com",

"password": "securepassword",

"firstName": "Max",

"lastName": "Mustermann",

"acceptsTerms": true,

"acceptsMarketing": false

}

Response:

{

"success": true,

"data": {

"user": {

"id": "usr\_123",

"email": "user@example.com",

"firstName": "Max",

"lastName": "Mustermann",

"subscription": "basic",

"createdAt": "2025-08-24T10:30:00Z"

},

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "rt\_456789"

}

}

**POST /auth/login**

Request:

{

"email": "user@example.com",

"password": "securepassword"

}

Response:

{

"success": true,

"data": {

"user": {...},

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"refreshToken": "rt\_456789",

"expiresIn": 3600

}

}

**2.2 Store/Chain Endpoints**

**GET /stores**

Query Parameters:

?lat=52.5200&lng=13.4050&radius=5000&chains=edeka,rewe,aldi

Response:

{

"success": true,

"data": {

"stores": [

{

"id": "store\_123",

"chainId": "edeka",

"chainName": "EDEKA",

"name": "EDEKA Hauptstraße",

"address": {

"street": "Hauptstraße 123",

"city": "Berlin",

"zipCode": "10115",

"country": "DE"

},

"coordinates": {

"lat": 52.5200,

"lng": 13.4050

},

"openingHours": {

"monday": "07:00-22:00",

"tuesday": "07:00-22:00",

"wednesday": "07:00-22:00",

"thursday": "07:00-22:00",

"friday": "07:00-22:00",

"saturday": "07:00-22:00",

"sunday": "08:00-20:00"

},

"distance": 1200,

"hasBeacon": true,

"activeFlashDeals": 3

}

]

}

}

**GET /chains**

Response:

{

"success": true,

"data": {

"chains": [

{

"id": "edeka",

"name": "EDEKA",

"logo": "https://api.flashfeed.de/assets/logos/edeka.png",

"color": "#005CA9",

"isActive": true,

"storeCount": 247

},

{

"id": "rewe",

"name": "REWE",

"logo": "https://api.flashfeed.de/assets/logos/rewe.png",

"color": "#CC071E",

"isActive": true,

"storeCount": 198

}

]

}

}

**2.3 Product & O􀆯er Endpoints**

**GET /o􀆯ers**

Query Parameters:

?chains=edeka,rewe&categories=dairy,meat&lat=52.5200&lng=13.4050&radius=5000

Response:

{

"success": true,

"data": {

"o􀆯ers": [

{

"id": "o􀆯er\_456",

"productId": "prod\_789",

"storeId": "store\_123",

"productName": "Vollmilch 3.5% 1L",

"brand": "Landliebe",

"category": "dairy",

"originalPrice": 1.29,

"discountedPrice": 0.99,

"discountPercentage": 23,

"validFrom": "2025-08-24T00:00:00Z",

"validUntil": "2025-08-30T23:59:59Z",

"isFlashDeal": false,

"chain": {

"id": "edeka",

"name": "EDEKA",

"logo": "https://api.flashfeed.de/assets/logos/edeka.png"

},

"store": {

"name": "EDEKA Hauptstraße",

"distance": 1200

}

}

],

"totalCount": 142,

"page": 1,

"pageSize": 20

}

}

**GET /flash-deals**

Response:

{

"success": true,

"data": {

"flashDeals": [

{

"id": "flash\_001",

"productId": "prod\_555",

"storeId": "store\_123",

"productName": "Bio-Äpfel 1kg",

"brand": "BioBio",

"category": "fruits",

"originalPrice": 3.49,

"discountedPrice": 1.75,

"discountPercentage": 50,

"validUntil": "2025-08-24T16:00:00Z",

"remainingTimeSeconds": 7200,

"isFlashDeal": true,

"urgencyLevel": "high", // high, medium, low

"chain": {

"id": "rewe",

"name": "REWE"

},

"store": {

"name": "REWE Alexanderplatz",

"address": "Alexanderplatz 5, 10178 Berlin",

"distance": 800

},

"shelfLocation": {

"aisle": "A3",

"shelf": "links",

"position": {

"x": 120,

"y": 340

}

}

}

]

}

}

**2.4 Categories Endpoint**

**GET /categories**

Response:

{

"success": true,

"data": {

"categories": [

{

"id": "dairy",

"name": "Milch & Milchprodukte",

"icon": "milk",

"color": "#4A90E2",

"o􀆯erCount": 23

},

{

"id": "meat",

"name": "Fleisch & Wurst",

"icon": "meat",

"color": "#E85D75",

"o􀆯erCount": 18

},

{

"id": "fruits",

"name": "Obst & Gemüse",

"icon": "apple",

"color": "#7ED321",

"o􀆯erCount": 31

},

{

"id": "bakery",

"name": "Backwaren",

"icon": "bread",

"color": "#D0A747",

"o􀆯erCount": 15

}

]

}

}

**2.5 User Subscription Endpoints**

**GET /user/subscription**

Response:

{

"success": true,

"data": {

"subscription": {

"plan": "premium", // basic, premium, family

"status": "active",

"startDate": "2025-08-01T00:00:00Z",

"endDate": "2025-09-01T00:00:00Z",

"autoRenew": true,

"allowedChains": ["edeka", "rewe", "aldi", "lidl"],

"maxChains": -1 // -1 = unlimited

},

"usage": {

"chainsUsed": 4,

"flashDealsUsed": 15,

"coinsEarned": 250,

"totalSavings": 45.67

}

}

}

**POST /user/subscription/chains**

Request:

{

"chainIds": ["edeka", "rewe", "aldi"]

}

Response:

{

"success": true,

"message": "Ketten-Auswahl erfolgreich aktualisiert"

}

**2.6 Shopping List Endpoints**

**GET /shopping-lists**

Response:

{

"success": true,

"data": {

"lists": [

{

"id": "list\_001",

"name": "Wocheneinkauf",

"items": [

{

"id": "item\_001",

"productName": "Milch 3.5% 1L",

"quantity": 2,

"unit": "Stück",

"category": "dairy",

"isCompleted": false,

"bestO􀆯er": {

"storeId": "store\_123",

"storeName": "EDEKA Hauptstraße",

"price": 0.99,

"originalPrice": 1.29,

"savings": 0.30

}

}

],

"createdAt": "2025-08-24T08:00:00Z",

"updatedAt": "2025-08-24T10:30:00Z",

"isShared": true,

"sharedWith": ["partner@example.com"]

}

]

}

}

**POST /shopping-lists**

Request:

{

"name": "Wocheneinkauf",

"items": [

{

"productName": "Milch 3.5% 1L",

"quantity": 2,

"unit": "Stück",

"category": "dairy"

}

]

}

Response:

{

"success": true,

"data": {

"listId": "list\_002",

"message": "Einkaufsliste erfolgreich erstellt"

}

}

**2.7 Store Floorplan Endpoints**

**GET /stores/{storeId}/floorplan**

Response:

{

"success": true,

"data": {

"floorplan": {

"id": "fp\_001",

"storeId": "store\_123",

"layout": {

"width": 800,

"height": 600,

"scale": "1px = 10cm"

},

"sections": [

{

"id": "dairy",

"name": "Milchprodukte",

"category": "dairy",

"coordinates": [

{"x": 100, "y": 200},

{"x": 200, "y": 200},

{"x": 200, "y": 300},

{"x": 100, "y": 300}

]

}

],

"flashDealLocations": [

{

"productId": "prod\_555",

"coordinates": {"x": 120, "y": 340},

"aisle": "A3",

"description": "Bio-Äpfel Regal links"

}

],

"entrance": {"x": 400, "y": 50},

"checkout": {"x": 350, "y": 550},

"lastUpdated": "2025-08-24T09:00:00Z"

}

}

}

**2.8 Notifications Endpoint**

**POST /notifications/register-device**

Request:

{

"deviceToken": "fcm\_token\_123456",

"platform": "web", // ios, android, web

"preferences": {

"flashDeals": true,

"nearbyO􀆯ers": true,

"expiringDeals": true,

"shoppingReminders": false

}

}

Response:

{

"success": true,

"message": "Gerät erfolgreich für Push-Notifications registriert"

}

**3. Mock Data Generation Endpoints**

**3.1 Generate Mock Flash Deals**

**POST /mock/generate-flash-deals**

Request:

{

"count": 5,

"storeIds": ["store\_123", "store\_456"],

"categories": ["dairy", "meat", "fruits"]

}

Response:

{

"success": true,

"data": {

"generated": 5,

"flashDeals": [

// Array of generated flash deals

]

}

}

**3.2 Simulate User Location**

**POST /mock/set-location**

Request:

{

"userId": "usr\_123",

"coordinates": {

"lat": 52.5200,

"lng": 13.4050

},

"locationName": "Berlin Mitte"

}

Response:

{

"success": true,

"message": "Mock-Standort gesetzt"

}

**4. Error Codes & Messages**

**4.1 HTTP Status Codes**

200 OK - Request successful

201 Created - Resource created successfully

400 Bad Request - Invalid request data

401 Unauthorized - Authentication required

403 Forbidden - Access denied

404 Not Found - Resource not found

422 Unprocessable Entity - Validation errors

429 Too Many Requests - Rate limit exceeded

500 Internal Server Error - Server error

503 Service Unavailable - Service temporarily unavailable

**4.2 Application Error Codes**

{

"success": false,

"message": "Validation failed",

"errors": [

{

"code": "INVALID\_EMAIL",

"field": "email",

"message": "E-Mail-Adresse ist ungültig"

},

{

"code": "SUBSCRIPTION\_LIMIT\_REACHED",

"field": "chains",

"message": "Maximale Anzahl Ketten erreicht. Upgrade auf Premium erforderlich."

}

]

}

**4.3 Common Error Codes**

AUTH\_001: Invalid credentials

AUTH\_002: Token expired

AUTH\_003: Account suspended

USER\_001: User not found

USER\_002: Email already exists

USER\_003: Invalid user data

STORE\_001: Store not found

STORE\_002: Store temporarily unavailable

STORE\_003: No beacon connection

OFFER\_001: O􀆯er expired

OFFER\_002: O􀆯er not available at location

OFFER\_003: Flash deal sold out

SUB\_001: Subscription expired

SUB\_002: Payment method invalid

SUB\_003: Chain limit reached

**5. Rate Limiting**

**5.1 Rate Limits per Endpoint**

Authentication: 5 requests/minute

User data: 60 requests/hour

O􀆯ers/Flash deals: 100 requests/hour

Store data: 50 requests/hour

Mock endpoints: 20 requests/hour

**5.2 Rate Limit Headers**

X-RateLimit-Limit: 100

X-RateLimit-Remaining: 95

X-RateLimit-Reset: 1635724800

**6. API Versioning**

**6.1 Versioning Strategy**

URL Versioning: /api/v1/, /api/v2/

Header Versioning: API-Version: v1

**6.2 Deprecation Policy**

- New version announcement: 30 days notice

- Support overlap: Minimum 90 days

- Breaking changes: Major version increment only

**7. WebSocket Events (Real-time Updates)**

**7.1 Connection**

const socket = io('wss://api.flashfeed.de', {

auth: {

token: 'Bearer jwt\_token\_here'

}

});

**7.2 Flash Deal Events**

// New flash deal available

socket.on('flash\_deal\_created', (data) => {

console.log('New flash deal:', data);

// data structure matches GET /flash-deals response

});

// Flash deal expiring soon

socket.on('flash\_deal\_expiring', (data) => {

console.log('Deal expiring in 30 minutes:', data);

});

// Flash deal expired

socket.on('flash\_deal\_expired', (data) => {

console.log('Deal expired:', data.dealId);

});

**7.3 Location-based Events**

// User entered store vicinity

socket.on('store\_proximity', (data) => {

console.log('Near store:', data.storeId);

// Trigger beacon simulation or floorplan download

});

// Beacon simulation trigger

socket.emit('simulate\_beacon\_entry', {

storeId: 'store\_123',

userId: 'usr\_123'

});

## Daten & Mock-System

### Datenbank Schema

**1. Technische Basis**

**1.1 Datenbank-Engine**

 **Entwicklung:** SQLite (lokale Web-Storage)

 **Produktion:** PostgreSQL 14+

 **Encoding:** UTF-8

 **Zeitzone:** UTC für alle Timestamps

**1.2 Naming Conventions**

-- Tabellen: plural, snake\_case

users, flash\_deals, shopping\_lists

-- Spalten: snake\_case

user\_id, created\_at, discount\_percentage

-- Indizes: idx\_tabelle\_spalte

idx\_users\_email, idx\_flash\_deals\_expires\_at

-- Foreign Keys: fk\_tabelle\_referenz

fk\_stores\_chain\_id, fk\_o􀆯ers\_product\_id

**2. Core Tables**

**2.1 Users**

CREATE TABLE users (

id VARCHAR(20) PRIMARY KEY DEFAULT ('usr\_' || generate\_random\_string(16)),

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

-- Subscription

subscription\_plan VARCHAR(20) DEFAULT 'basic', -- basic, premium, family

subscription\_status VARCHAR(20) DEFAULT 'active', -- active, expired, cancelled

subscription\_start DATE,

subscription\_end DATE,

auto\_renew BOOLEAN DEFAULT true,

-- Preferences

preferred\_chains TEXT[], -- ['edeka', 'rewe', 'aldi']

notification\_preferences JSONB DEFAULT '{

"flash\_deals": true,

"nearby\_o􀆯ers": true,

"expiring\_deals": true,

"shopping\_reminders": false

}',

-- Location

last\_known\_lat DECIMAL(10, 8),

last\_known\_lng DECIMAL(11, 8),

preferred\_radius INTEGER DEFAULT 5000, -- meters

-- Gamification

total\_coins INTEGER DEFAULT 0,

total\_savings DECIMAL(10, 2) DEFAULT 0.00,

user\_level INTEGER DEFAULT 1,

-- Timestamps

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

last\_login\_at TIMESTAMP,

-- GDPR

terms\_accepted\_at TIMESTAMP NOT NULL,

marketing\_consent BOOLEAN DEFAULT false,

data\_processing\_consent BOOLEAN DEFAULT true

);

CREATE INDEX idx\_users\_email ON users(email);

CREATE INDEX idx\_users\_subscription\_status ON users(subscription\_status);

CREATE INDEX idx\_users\_location ON users(last\_known\_lat, last\_known\_lng);

**2.2 Chains (Handelsketten)**

CREATE TABLE chains (

id VARCHAR(20) PRIMARY KEY, -- 'edeka', 'rewe', 'aldi'

name VARCHAR(100) NOT NULL,

display\_name VARCHAR(100) NOT NULL,

logo\_url VARCHAR(255),

primary\_color VARCHAR(7), -- #005CA9

-- Business Info

website VARCHAR(255),

customer\_service\_phone VARCHAR(20),

-- Platform Status

is\_active BOOLEAN DEFAULT true,

api\_integration\_status VARCHAR(20) DEFAULT 'mock', -- mock, testing, live

-- Statistics (cached)

total\_stores INTEGER DEFAULT 0,

active\_o􀆯ers INTEGER DEFAULT 0,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

INSERT INTO chains VALUES

('edeka', 'EDEKA', 'EDEKA', '/assets/logos/edeka.png', '#005CA9', 'https://www.edeka.de',

'0800-3335225', true, 'mock', 0, 0, NOW(), NOW()),

('rewe', 'REWE', 'REWE', '/assets/logos/rewe.png', '#CC071E', 'https://www.rewe.de',

'0221-1491', true, 'mock', 0, 0, NOW(), NOW()),

('aldi', 'ALDI SÜD', 'ALDI', '/assets/logos/aldi.png', '#00549F', 'https://www.aldi-sued.de',

'0800-2534638', true, 'mock', 0, 0, NOW(), NOW()),

('lidl', 'LIDL', 'LIDL', '/assets/logos/lidl.png', '#0050AA', 'https://www.lidl.de', '0800-

4353535', true, 'mock', 0, 0, NOW(), NOW()),

('netto\_schwarz', 'NETTO', 'Netto', '/assets/logos/netto-schwarz.png', '#FFD100',

'https://www.netto-online.de', '0800-2000015', true, 'mock', 0, 0, NOW(), NOW());

**2.3 Stores (Filialen)**

CREATE TABLE stores (

id VARCHAR(20) PRIMARY KEY DEFAULT ('store\_' || generate\_random\_string(12)),

chain\_id VARCHAR(20) NOT NULL REFERENCES chains(id),

-- Basic Info

name VARCHAR(200) NOT NULL,

store\_number VARCHAR(20), -- Internal chain store number

-- Address

street VARCHAR(200) NOT NULL,

house\_number VARCHAR(10),

zip\_code VARCHAR(10) NOT NULL,

city VARCHAR(100) NOT NULL,

country VARCHAR(2) DEFAULT 'DE',

-- Coordinates

latitude DECIMAL(10, 8) NOT NULL,

longitude DECIMAL(11, 8) NOT NULL,

-- Opening Hours (JSON format for flexibility)

opening\_hours JSONB DEFAULT '{

"monday": "07:00-22:00",

"tuesday": "07:00-22:00",

"wednesday": "07:00-22:00",

"thursday": "07:00-22:00",

"friday": "07:00-22:00",

"saturday": "07:00-22:00",

"sunday": "08:00-20:00"

}',

-- Features

has\_beacon BOOLEAN DEFAULT false,

has\_parking BOOLEAN DEFAULT true,

is\_accessible BOOLEAN DEFAULT true,

-- Status

is\_active BOOLEAN DEFAULT true,

temporary\_closed BOOLEAN DEFAULT false,

-- Statistics

avg\_rating DECIMAL(3, 2) DEFAULT 0.00,

total\_reviews INTEGER DEFAULT 0,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_stores\_chain\_id ON stores(chain\_id);

CREATE INDEX idx\_stores\_location ON stores(latitude, longitude);

CREATE INDEX idx\_stores\_city ON stores(city);

CREATE INDEX idx\_stores\_active ON stores(is\_active);

**2.4 Product Categories**

CREATE TABLE categories (

id VARCHAR(20) PRIMARY KEY, -- 'dairy', 'meat', 'fruits'

name VARCHAR(100) NOT NULL,

display\_name VARCHAR(100) NOT NULL,

description TEXT,

-- UI

icon\_name VARCHAR(50), -- 'milk', 'meat', 'apple'

color VARCHAR(7), -- #4A90E2

sort\_order INTEGER DEFAULT 0,

-- Mapping für verschiedene Handelsketten

chain\_category\_mappings JSONB DEFAULT '{}', -- {"edeka": "Molkereiprodukte", "rewe":

"Milch & Käse"}

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

INSERT INTO categories VALUES

('dairy', 'Dairy', 'Milch & Milchprodukte', 'Milch, Joghurt, Käse, Butter', 'milk', '#4A90E2', 1,

'{"edeka": "Molkereiprodukte", "rewe": "Milch & Käse", "aldi": "Milchprodukte"}', true,

NOW()),

('meat', 'Meat', 'Fleisch & Wurst', 'Fleisch, Wurst, Geflügel', 'meat', '#E85D75', 2, '{"edeka":

"Fleisch & Wurst", "rewe": "Fleisch", "aldi": "Fleischwaren"}', true, NOW()),

('fruits', 'Fruits', 'Obst & Gemüse', 'Frisches Obst und Gemüse', 'apple', '#7ED321', 3,

'{"edeka": "Obst & Gemüse", "rewe": "Frische", "aldi": "Obst/Gemüse"}', true, NOW()),

('bakery', 'Bakery', 'Backwaren', 'Brot, Brötchen, Gebäck', 'bread', '#D0A747', 4, '{"edeka":

"Bäckerei", "rewe": "Backshop", "aldi": "Backwaren"}', true, NOW()),

('beverages', 'Beverages', 'Getränke', 'Wasser, Säfte, Softdrinks', 'co􀆯ee', '#6BB6FF', 5,

'{"edeka": "Getränke", "rewe": "Getränke", "aldi": "Getränke"}', true, NOW());

**2.5 Products (Mock-Produktdatenbank)**

CREATE TABLE products (

id VARCHAR(20) PRIMARY KEY DEFAULT ('prod\_' || generate\_random\_string(12)),

category\_id VARCHAR(20) NOT NULL REFERENCES categories(id),

-- Product Info

name VARCHAR(200) NOT NULL,

brand VARCHAR(100),

description TEXT,

-- Specifications

size VARCHAR(50), -- '1L', '500g', '250ml'

unit VARCHAR(20), -- 'Stück', 'kg', 'L'

ean\_code VARCHAR(20), -- Barcode for scanning

-- Base Price (varies by store/chain)

base\_price\_cents INTEGER, -- Price in cents: 129 = 1.29€

-- Categorization

tags TEXT[], -- ['bio', 'regional', 'lactose-free']

allergens TEXT[], -- ['milk', 'gluten', 'nuts']

-- Availability

seasonal BOOLEAN DEFAULT false,

available\_from DATE,

available\_until DATE,

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_products\_category\_id ON products(category\_id);

CREATE INDEX idx\_products\_brand ON products(brand);

CREATE INDEX idx\_products\_name ON products(name);

**2.6 Regular O􀆯ers (Prospekt-Angebote)**

CREATE TABLE o􀆯ers (

id VARCHAR(20) PRIMARY KEY DEFAULT ('o􀆯er\_' || generate\_random\_string(12)),

product\_id VARCHAR(20) NOT NULL REFERENCES products(id),

store\_id VARCHAR(20) NOT NULL REFERENCES stores(id),

chain\_id VARCHAR(20) NOT NULL REFERENCES chains(id),

-- Pricing

original\_price\_cents INTEGER NOT NULL,

discounted\_price\_cents INTEGER NOT NULL,

discount\_percentage INTEGER, -- Calculated field

-- Validity

valid\_from DATE NOT NULL,

valid\_until DATE NOT NULL,

-- Source & Type

source VARCHAR(50) DEFAULT 'prospekt', -- prospekt, flash, manual

o􀆯er\_type VARCHAR(20) DEFAULT 'regular', -- regular, weekly, seasonal

-- Metadata

o􀆯er\_text TEXT, -- "2 für 1", "Statt 2.99 nur 1.99"

conditions TEXT, -- "Nur solange Vorrat reicht"

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_o􀆯ers\_store\_id ON o􀆯ers(store\_id);

CREATE INDEX idx\_o􀆯ers\_valid\_until ON o􀆯ers(valid\_until);

CREATE INDEX idx\_o􀆯ers\_discount\_percentage ON o􀆯ers(discount\_percentage);

CREATE INDEX idx\_o􀆯ers\_active ON o􀆯ers(is\_active);

**2.7 Flash Deals (Echtzeit-Rabatte)**

CREATE TABLE flash\_deals (

id VARCHAR(20) PRIMARY KEY DEFAULT ('flash\_' || generate\_random\_string(12)),

product\_id VARCHAR(20) NOT NULL REFERENCES products(id),

store\_id VARCHAR(20) NOT NULL REFERENCES stores(id),

chain\_id VARCHAR(20) NOT NULL REFERENCES chains(id),

-- Pricing

original\_price\_cents INTEGER NOT NULL,

flash\_price\_cents INTEGER NOT NULL,

discount\_percentage INTEGER, -- Calculated: ((original - flash) / original) \* 100

-- Time Constraints

starts\_at TIMESTAMP NOT NULL,

expires\_at TIMESTAMP NOT NULL,

remaining\_seconds INTEGER, -- Calculated field for frontend

-- Urgency & Stock

urgency\_level VARCHAR(10) DEFAULT 'medium', -- low, medium, high

estimated\_stock INTEGER, -- How many items available

max\_per\_customer INTEGER DEFAULT 5,

-- Location in Store

shelf\_location JSONB, -- {"aisle": "A3", "shelf": "links", "x": 120, "y": 340}

-- Performance Tracking

views INTEGER DEFAULT 0,

clicks INTEGER DEFAULT 0,

redemptions INTEGER DEFAULT 0,

-- ERP Integration

erp\_trigger\_rule TEXT, -- "MHD < 2 days", "Stock > 50 items"

auto\_generated BOOLEAN DEFAULT false,

-- Status

status VARCHAR(20) DEFAULT 'active', -- active, expired, sold\_out, cancelled

is\_featured BOOLEAN DEFAULT false,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_flash\_deals\_expires\_at ON flash\_deals(expires\_at);

CREATE INDEX idx\_flash\_deals\_store\_id ON flash\_deals(store\_id);

CREATE INDEX idx\_flash\_deals\_status ON flash\_deals(status);

CREATE INDEX idx\_flash\_deals\_urgency ON flash\_deals(urgency\_level);

**2.8 Shopping Lists**

CREATE TABLE shopping\_lists (

id VARCHAR(20) PRIMARY KEY DEFAULT ('list\_' || generate\_random\_string(12)),

user\_id VARCHAR(20) NOT NULL REFERENCES users(id),

name VARCHAR(200) NOT NULL DEFAULT 'Einkaufsliste',

description TEXT,

-- Sharing

is\_shared BOOLEAN DEFAULT false,

shared\_with TEXT[], -- Email addresses

-- Status

is\_completed BOOLEAN DEFAULT false,

completed\_at TIMESTAMP,

-- Statistics

total\_items INTEGER DEFAULT 0,

completed\_items INTEGER DEFAULT 0,

estimated\_total\_cents INTEGER DEFAULT 0,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_shopping\_lists\_user\_id ON shopping\_lists(user\_id);

CREATE INDEX idx\_shopping\_lists\_updated\_at ON shopping\_lists(updated\_at);

**2.9 Shopping List Items**

CREATE TABLE shopping\_list\_items (

id VARCHAR(20) PRIMARY KEY DEFAULT ('item\_' || generate\_random\_string(12)),

list\_id VARCHAR(20) NOT NULL REFERENCES shopping\_lists(id) ON DELETE

CASCADE,

-- Product Reference (optional - can be free text)

product\_id VARCHAR(20) REFERENCES products(id),

product\_name VARCHAR(200) NOT NULL, -- User can override product name

-- Quantity

quantity DECIMAL(8, 3) DEFAULT 1.0,

unit VARCHAR(20) DEFAULT 'Stück',

-- Category (for organization)

category\_id VARCHAR(20) REFERENCES categories(id),

-- Status

is\_completed BOOLEAN DEFAULT false,

completed\_at TIMESTAMP,

-- Best O􀆯er Tracking

best\_o􀆯er\_id VARCHAR(20), -- References o􀆯ers(id) or flash\_deals(id)

best\_o􀆯er\_type VARCHAR(10), -- 'o􀆯er' or 'flash'

best\_price\_cents INTEGER,

best\_store\_id VARCHAR(20) REFERENCES stores(id),

-- Priority

priority INTEGER DEFAULT 5, -- 1 = lowest, 10 = highest

notes TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_shopping\_list\_items\_list\_id ON shopping\_list\_items(list\_id);

CREATE INDEX idx\_shopping\_list\_items\_category\_id ON

shopping\_list\_items(category\_id);

CREATE INDEX idx\_shopping\_list\_items\_completed ON

shopping\_list\_items(is\_completed);

**3. Supporting Tables**

**3.1 User Subscriptions (Historie)**

CREATE TABLE user\_subscriptions (

id VARCHAR(20) PRIMARY KEY DEFAULT ('sub\_' || generate\_random\_string(12)),

user\_id VARCHAR(20) NOT NULL REFERENCES users(id),

plan VARCHAR(20) NOT NULL, -- basic, premium, family

status VARCHAR(20) NOT NULL, -- active, expired, cancelled, refunded

-- Pricing

price\_cents INTEGER NOT NULL,

currency VARCHAR(3) DEFAULT 'EUR',

-- Billing Period

billing\_cycle VARCHAR(20), -- monthly, yearly

starts\_at DATE NOT NULL,

ends\_at DATE NOT NULL,

-- Payment

payment\_method VARCHAR(50), -- stripe\_card, paypal, etc.

payment\_reference VARCHAR(100), -- External payment ID

-- Permissions

max\_chains INTEGER DEFAULT 1, -- -1 = unlimited

features JSONB DEFAULT '{}', -- Additional feature flags

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

cancelled\_at TIMESTAMP,

refunded\_at TIMESTAMP

);

CREATE INDEX idx\_user\_subscriptions\_user\_id ON user\_subscriptions(user\_id);

CREATE INDEX idx\_user\_subscriptions\_status ON user\_subscriptions(status);

CREATE INDEX idx\_user\_subscriptions\_ends\_at ON user\_subscriptions(ends\_at);

**3.2 Store Floorplans**

CREATE TABLE store\_floorplans (

id VARCHAR(20) PRIMARY KEY DEFAULT ('fp\_' || generate\_random\_string(12)),

store\_id VARCHAR(20) NOT NULL REFERENCES stores(id),

-- Layout Specifications

layout\_width INTEGER NOT NULL, -- pixels

layout\_height INTEGER NOT NULL, -- pixels

scale\_info VARCHAR(100), -- "1px = 10cm"

-- SVG/JSON Data

layout\_data JSONB NOT NULL, -- Complete floorplan as JSON

svg\_content TEXT, -- Optional SVG representation

-- Sections (Product Areas)

sections JSONB DEFAULT '[]', -- [{"id": "dairy", "name": "Milchprodukte", "coordinates":

[...]}]

-- Fixed Elements

entrance\_coordinates JSONB, -- {"x": 400, "y": 50}

checkout\_coordinates JSONB, -- [{"x": 350, "y": 550}, {"x": 400, "y": 550}]

emergency\_exits JSONB DEFAULT '[]',

-- Version Control

version INTEGER DEFAULT 1,

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

created\_by VARCHAR(20) -- Sta􀆯 member who created/updated

);

CREATE INDEX idx\_store\_floorplans\_store\_id ON store\_floorplans(store\_id);

CREATE INDEX idx\_store\_floorplans\_active ON store\_floorplans(is\_active);

**3.3 User Activity Log**

CREATE TABLE user\_activities (

id VARCHAR(20) PRIMARY KEY DEFAULT ('act\_' || generate\_random\_string(12)),

user\_id VARCHAR(20) NOT NULL REFERENCES users(id),

-- Activity Details

activity\_type VARCHAR(50) NOT NULL, -- 'flash\_deal\_viewed', 'o􀆯er\_clicked',

'store\_visited'

entity\_type VARCHAR(50), -- 'flash\_deal', 'o􀆯er', 'store', 'product'

entity\_id VARCHAR(20), -- ID of the related entity

-- Context

store\_id VARCHAR(20) REFERENCES stores(id),

chain\_id VARCHAR(20) REFERENCES chains(id),

-- Location Context

user\_lat DECIMAL(10, 8),

user\_lng DECIMAL(11, 8),

-- Metadata

metadata JSONB DEFAULT '{}', -- Additional context data

-- Device Info

device\_type VARCHAR(20), -- 'mobile', 'tablet', 'desktop'

user\_agent TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_user\_activities\_user\_id ON user\_activities(user\_id);

CREATE INDEX idx\_user\_activities\_type ON user\_activities(activity\_type);

CREATE INDEX idx\_user\_activities\_created\_at ON user\_activities(created\_at);

**3.4 Push Notifications**

CREATE TABLE push\_notifications (

id VARCHAR(20) PRIMARY KEY DEFAULT ('notif\_' || generate\_random\_string(12)),

user\_id VARCHAR(20) NOT NULL REFERENCES users(id),

-- Notification Content

title VARCHAR(200) NOT NULL,

body TEXT NOT NULL,

-- Notification Type

notification\_type VARCHAR(50) NOT NULL, -- 'flash\_deal', 'nearby\_o􀆯er',

'expiring\_deal'

-- Related Entities

related\_entity\_type VARCHAR(50), -- 'flash\_deal', 'o􀆯er'

related\_entity\_id VARCHAR(20),

-- Delivery

scheduled\_for TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

sent\_at TIMESTAMP,

delivered\_at TIMESTAMP,

clicked\_at TIMESTAMP,

-- Device Targeting

device\_token VARCHAR(255),

platform VARCHAR(20), -- 'ios', 'android', 'web'

-- Status

status VARCHAR(20) DEFAULT 'pending', -- pending, sent, delivered, clicked, failed

error\_message TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_push\_notifications\_user\_id ON push\_notifications(user\_id);

CREATE INDEX idx\_push\_notifications\_scheduled\_for ON

push\_notifications(scheduled\_for);

CREATE INDEX idx\_push\_notifications\_status ON push\_notifications(status);

**4. Views & Computed Fields**

**4.1 Active Flash Deals View**

CREATE VIEW active\_flash\_deals AS

SELECT

fd.\*,

p.name as product\_name,

p.brand,

p.size,

c.name as category\_name,

s.name as store\_name,

s.city,

ch.name as chain\_name,

ch.logo\_url as chain\_logo,

EXTRACT(EPOCH FROM (fd.expires\_at - NOW()))::INTEGER as remaining\_seconds

FROM flash\_deals fd

JOIN products p ON fd.product\_id = p.id

JOIN categories c ON p.category\_id = c.id

JOIN stores s ON fd.store\_id = s.id

JOIN chains ch ON fd.chain\_id = ch.id

WHERE fd.status = 'active'

AND fd.expires\_at > NOW()

AND s.is\_active = true

AND ch.is\_active = true;

**4.2 User Savings Summary View**

CREATE VIEW user\_savings\_summary AS

SELECT

u.id as user\_id,

u.first\_name,

u.last\_name,

COUNT(DISTINCT ua.id) FILTER (WHERE ua.activity\_type = 'flash\_deal\_redeemed') as

flash\_deals\_used,

COUNT(DISTINCT ua.id) FILTER (WHERE ua.activity\_type = 'o􀆯er\_redeemed') as

regular\_o􀆯ers\_used,

COALESCE(SUM((ua.metadata->>'savings\_cents')::INTEGER), 0) as

total\_savings\_cents,

u.total\_coins,

u.user\_level

FROM users u

LEFT JOIN user\_activities ua ON u.id = ua.user\_id

WHERE ua.activity\_type IN ('flash\_deal\_redeemed', 'o􀆯er\_redeemed')

GROUP BY u.id, u.first\_name, u.last\_name, u.total\_coins, u.user\_level;

**5. Triggers & Functions**

**5.1 Auto-Update Timestamps**

CREATE OR REPLACE FUNCTION update\_updated\_at\_column()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = CURRENT\_TIMESTAMP;

RETURN NEW;

END;

$$ language 'plpgsql';

CREATE TRIGGER update\_users\_updated\_at BEFORE UPDATE ON users

FOR EACH ROW EXECUTE FUNCTION update\_updated\_at\_column();

**5.2 Calculate Discount Percentage**

CREATE OR REPLACE FUNCTION calculate\_discount\_percentage()

RETURNS TRIGGER AS $$

BEGIN

NEW.discount\_percentage = ROUND(

((NEW.original\_price\_cents - NEW.discounted\_price\_cents)::DECIMAL /

NEW.original\_price\_cents::DECIMAL) \* 100

)::INTEGER;

RETURN NEW;

END;

$$ language 'plpgsql';

CREATE TRIGGER calculate\_flash\_deals\_discount BEFORE INSERT OR UPDATE ON

flash\_deals

FOR EACH ROW EXECUTE FUNCTION calculate\_discount\_percentage();

CREATE TRIGGER calculate\_o􀆯ers\_discount BEFORE INSERT OR UPDATE ON o􀆯ers

FOR EACH ROW EXECUTE FUNCTION calculate\_discount\_percentage();

**6. Sample Data Queries**

**6.1 Insert Mock Flash Deal**

INSERT INTO flash\_deals (

product\_id, store\_id, chain\_id,

original\_price\_cents, flash\_price\_cents,

starts\_at, expires\_at,

urgency\_level, estimated\_stock,

shelf\_location, auto\_generated

) VALUES (

'prod\_001', 'store\_123', 'edeka',

349, 175, -- 3.49€ -> 1.75€ (50% o􀆯)

NOW(), NOW() + INTERVAL '2 hours',

'high', 15,

'{"aisle": "A3", "shelf": "links", "x": 120, "y": 340}',

true

);

**6.2 Find Nearby O􀆯ers**

SELECT

o.\*,

p.name as product\_name,

s.name as store\_name,

ST\_Distance(

ST\_Point(s.longitude, s.latitude),

ST\_Point(13.4050, 52.5200) -- User location

) \* 111320 as distance\_meters

FROM o􀆯ers o

JOIN products p ON o.product\_id = p.id

JOIN stores s ON o.store\_id = s.id

WHERE o.is\_active = true

AND o.valid\_until >= CURRENT\_DATE

AND ST\_DWithin(

ST\_Point(s.longitude, s.latitude),

ST\_Point(13.4050, 52.5200), -- User location

0.045 -- ~5km radius

)

ORDER BY distance\_meters, o.discount\_percentage DESC;

### Mock-Daten Spezifikationen

**1. Mock-Daten Architektur**

**1.1 Datenstruktur**

// Hauptstruktur für Mock-Data-Manager

const mockDataStructure = {

chains: [], // 10 deutsche LEH-Ketten

stores: [], // 50 Filialen (5 pro Kette)

categories: [], // 12 Produktkategorien

products: [], // 500 realistische Produkte

o􀆯ers: [], // 200 reguläre Angebote

flashDeals: [], // 15-20 aktive Flash-Deals

users: [], // 5 Test-User-Profile

shoppingLists: [], // 10 Einkaufslisten

floorplans: [] // 15 Standard-Lagepläne

};

## Implementation Details

### Flutter Projektstruktur & Implementation

**1. Vollständige Projektstruktur**

flashfeed\_prototype/

├── README.md

├── pubspec.yaml

├── analysis\_options.yaml

├── web/

│ ├── index.html

│ ├── manifest.json

│ └── icons/

│ ├── Icon-192.png

│ ├── Icon-512.png

│ └── favicon.png

├── assets/

│ ├── images/

│ │ ├── logos/

│ │ │ ├── edeka.png

│ │ │ ├── rewe.png

│ │ │ ├── aldi.png

│ │ │ ├── lidl.png

│ │ │ └── netto-schwarz.png

│ │ └── app\_logo.png

│ └── data/

│ ├── mock\_chains.json

│ ├── mock\_stores.json

│ ├── mock\_products.json

│ └── floorplans/

│ ├── store\_001.json

│ ├── store\_002.json

│ └── ...

├── lib/

│ ├── main.dart

│ ├── app.dart

│ ├── core/

│ │ ├── constants/

│ │ │ ├── app\_colors.dart

│ │ │ ├── app\_text\_styles.dart

│ │ │ ├── app\_dimensions.dart

│ │ │ └── app\_strings.dart

│ │ ├── theme/

│ │ │ └── app\_theme.dart

│ │ ├── utils/

│ │ │ ├── date\_utils.dart

│ │ │ ├── price\_utils.dart

│ │ │ └── location\_utils.dart

│ │ └── services/

│ │ ├── storage\_service.dart

│ │ ├── notification\_service.dart

│ │ ├── location\_service.dart

│ │ └── mock\_data\_service.dart

│ ├── data/

│ │ ├── models/

│ │ │ ├── chain.dart

│ │ │ ├── store.dart

│ │ │ ├── product.dart

│ │ │ ├── o􀆯er.dart

│ │ │ ├── flash\_deal.dart

│ │ │ ├── user.dart

│ │ │ ├── shopping\_list.dart

│ │ │ ├── category.dart

│ │ │ └── floorplan.dart

│ │ ├── repositories/

│ │ │ ├── chain\_repository.dart

│ │ │ ├── store\_repository.dart

│ │ │ ├── o􀆯er\_repository.dart

│ │ │ ├── flash\_deal\_repository.dart

│ │ │ ├── user\_repository.dart

│ │ │ └── shopping\_list\_repository.dart

│ │ └── datasources/

│ │ ├── local\_storage\_datasource.dart

│ │ └── mock\_data\_generator.dart

│ ├── presentation/

│ │ ├── blocs/

│ │ │ ├── main\_navigation/

│ │ │ │ ├── main\_navigation\_bloc.dart

│ │ │ │ ├── main\_navigation\_event.dart

│ │ │ │ └── main\_navigation\_state.dart

│ │ │ ├── o􀆯ers/

│ │ │ │ ├── o􀆯ers\_bloc.dart

│ │ │ │ ├── o􀆯ers\_event.dart

│ │ │ │ └── o􀆯ers\_state.dart

│ │ │ ├── flash\_deals/

│ │ │ │ ├── flash\_deals\_bloc.dart

│ │ │ │ ├── flash\_deals\_event.dart

│ │ │ │ └── flash\_deals\_state.dart

│ │ │ ├── map/

│ │ │ │ ├── map\_bloc.dart

│ │ │ │ ├── map\_event.dart

│ │ │ │ └── map\_state.dart

│ │ │ └── shopping\_list/

│ │ │ ├── shopping\_list\_bloc.dart

│ │ │ ├── shopping\_list\_event.dart

│ │ │ └── shopping\_list\_state.dart

│ │ ├── screens/

│ │ │ ├── main\_screen.dart

│ │ │ ├── o􀆯ers\_screen.dart

│ │ │ ├── map\_screen.dart

│ │ │ ├── flash\_deals\_screen.dart

│ │ │ ├── settings\_screen.dart

│ │ │ └── floorplan\_screen.dart

│ │ └── widgets/

│ │ ├── common/

│ │ │ ├── app\_header.dart

│ │ │ ├── loading\_spinner.dart

│ │ │ ├── error\_message.dart

│ │ │ └── countdown\_timer.dart

│ │ ├── o􀆯ers/

│ │ │ ├── chain\_selector.dart

│ │ │ ├── category\_grid.dart

│ │ │ ├── o􀆯er\_card.dart

│ │ │ └── chain\_icon.dart

│ │ ├── map/

│ │ │ ├── store\_map.dart

│ │ │ ├── radius\_filter.dart

│ │ │ └── store\_pin.dart

│ │ ├── flash\_deals/

│ │ │ ├── flash\_deal\_card.dart

│ │ │ ├── urgency\_badge.dart

│ │ │ └── deal\_countdown.dart

│ │ └── floorplan/

│ │ ├── floorplan\_widget.dart

│ │ ├── product\_marker.dart

│ │ └── store\_layout.dart

│ └── generated/

│ └── assets.dart

├── .github/

│ └── workflows/

│ └── deploy.yml

└── docs/

└── DEPLOYMENT.md

**2. pubspec.yaml Konfiguration**

name: flashfeed\_prototype

description: FlashFeed prototype for academic project

version: 1.0.0+1

environment:

sdk: '>=3.1.0 <4.0.0'

flutter: ">=3.13.0"

dependencies:

flutter:

sdk: flutter

# State Management

flutter\_bloc: ^8.1.3

equatable: ^2.0.5

# Storage

shared\_preferences: ^2.2.2

# HTTP & JSON

http: ^1.1.0

json\_annotation: ^4.8.1

# Maps

google\_maps\_flutter\_web: ^0.5.4+2

geolocator: ^10.1.0

# UI & Icons

lucide\_icons: ^0.263.0

flutter\_svg: ^2.0.7

# Utilities

intl: ^0.18.1

uuid: ^4.1.0

# Notifications (Web)

flutter\_local\_notifications: ^16.1.0

# Assets

flutter\_gen: ^5.3.2

dev\_dependencies:

flutter\_test:

sdk: flutter

flutter\_lints: ^3.0.0

build\_runner: ^2.4.7

json\_serializable: ^6.7.1

flutter\_gen\_runner: ^5.3.2

flutter:

uses-material-design: true

assets:

- assets/images/

- assets/images/logos/

- assets/data/

- assets/data/floorplans/

fonts:

- family: Roboto

fonts:

- asset: fonts/Roboto-Regular.ttf

- asset: fonts/Roboto-Medium.ttf

weight: 500

- asset: fonts/Roboto-Bold.ttf

weight: 700

flutter\_gen:

assets:

enabled: true

package\_parameter\_enabled: true

**3. Haupt-Implementierungsdateien**

**3.1 main.dart**

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'app.dart';

import 'core/services/storage\_service.dart';

import 'core/services/mock\_data\_service.dart';

import 'data/repositories/chain\_repository.dart';

import 'data/repositories/o􀆯er\_repository.dart';

import 'data/repositories/flash\_deal\_repository.dart';

import 'data/repositories/user\_repository.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

// Initialize services

final sharedPreferences = await SharedPreferences.getInstance();

final storageService = StorageService(sharedPreferences);

final mockDataService = MockDataService();

// Initialize repositories

final chainRepository = ChainRepository(storageService);

final o􀆯erRepository = O􀆯erRepository(storageService);

final flashDealRepository = FlashDealRepository(storageService);

final userRepository = UserRepository(storageService);

// Generate initial mock data if not exists

await mockDataService.initializeMockData(storageService);

runApp(FlashFeedApp(

chainRepository: chainRepository,

o􀆯erRepository: o􀆯erRepository,

flashDealRepository: flashDealRepository,

userRepository: userRepository,

mockDataService: mockDataService,

));

}

**3.2 app.dart**

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'core/theme/app\_theme.dart';

import 'data/repositories/chain\_repository.dart';

import 'data/repositories/o􀆯er\_repository.dart';

import 'data/repositories/flash\_deal\_repository.dart';

import 'data/repositories/user\_repository.dart';

import 'core/services/mock\_data\_service.dart';

import 'presentation/blocs/main\_navigation/main\_navigation\_bloc.dart';

import 'presentation/blocs/o􀆯ers/o􀆯ers\_bloc.dart';

import 'presentation/blocs/flash\_deals/flash\_deals\_bloc.dart';

import 'presentation/blocs/map/map\_bloc.dart';

import 'presentation/screens/main\_screen.dart';

class FlashFeedApp extends StatelessWidget {

final ChainRepository chainRepository;

final O􀆯erRepository o􀆯erRepository;

final FlashDealRepository flashDealRepository;

final UserRepository userRepository;

final MockDataService mockDataService;

const FlashFeedApp({

super.key,

required this.chainRepository,

required this.o􀆯erRepository,

required this.flashDealRepository,

required this.userRepository,

required this.mockDataService,

});

@override

Widget build(BuildContext context) {

return MultiBlocProvider(

providers: [

BlocProvider<MainNavigationBloc>(

create: (context) => MainNavigationBloc(),

),

BlocProvider<O􀆯ersBloc>(

create: (context) => O􀆯ersBloc(

chainRepository: chainRepository,

o􀆯erRepository: o􀆯erRepository,

)..add(LoadO􀆯ersEvent()),

),

BlocProvider<FlashDealsBloc>(

create: (context) => FlashDealsBloc(

flashDealRepository: flashDealRepository,

mockDataService: mockDataService,

)..add(LoadFlashDealsEvent()),

),

BlocProvider<MapBloc>(

create: (context) => MapBloc(

chainRepository: chainRepository,

),

),

],

child: MaterialApp(

title: 'FlashFeed Prototype',

theme: AppTheme.lightTheme,

home: const MainScreen(),

debugShowCheckedModeBanner: false,

),

);

}

}

**3.3 Core Constants - app\_colors.dart**

import 'package:flutter/material.dart';

class AppColors {

// Primary Colors

static const Color primaryGreen = Color(0xFF2E8B57); // SeaGreen

static const Color primaryRed = Color(0xFFDC143C); // Crimson

static const Color primaryBlue = Color(0xFF1E90FF); // DodgerBlue

// Secondary Colors

static const Color secondaryLightGreen = Color(0xFF90EE90);

static const Color secondaryOrange = Color(0xFFFF6347);

static const Color secondaryGray = Color(0xFF708090);

// Neutral Colors

static const Color backgroundLight = Color(0xFFFAFAFA);

static const Color backgroundDark = Color(0xFF2C2C2C);

static const Color textPrimary = Color(0xFF333333);

static const Color textSecondary = Color(0xFF666666);

static const Color textWhite = Color(0xFFFFFFFF);

// Status Colors

static const Color success = Color(0xFF28A745);

static const Color warning = Color(0xFFFFC107);

static const Color error = Color(0xFFDC3545);

static const Color info = Color(0xFF17A2B8);

// Chain Colors

static const Color edekaBlue = Color(0xFF005CA9);

static const Color reweRed = Color(0xFFCC071E);

static const Color aldiBlue = Color(0xFF00549F);

static const Color lidlBlue = Color(0xFF0050AA);

static const Color nettoYellow = Color(0xFFFFD100);

}

**3.4 Data Models - flash\_deal.dart**

import 'package:json\_annotation/json\_annotation.dart';

import 'package:equatable/equatable.dart';

part 'flash\_deal.g.dart';

@JsonSerializable()

class FlashDeal extends Equatable {

final String id;

final String productId;

final String storeId;

final String chainId;

final String productName;

final String brand;

final String storeName;

final String chainName;

final int originalPriceCents;

final int flashPriceCents;

final int discountPercentage;

final DateTime startsAt;

final DateTime expiresAt;

final int remainingSeconds;

final String urgencyLevel; // 'low', 'medium', 'high'

final int estimatedStock;

final int maxPerCustomer;

final ShelfLocation? shelfLocation;

final String status; // 'active', 'expired', 'sold\_out'

final bool isFeatured;

final String? erpTriggerRule;

const FlashDeal({

required this.id,

required this.productId,

required this.storeId,

required this.chainId,

required this.productName,

required this.brand,

required this.storeName,

required this.chainName,

required this.originalPriceCents,

required this.flashPriceCents,

required this.discountPercentage,

required this.startsAt,

required this.expiresAt,

required this.remainingSeconds,

required this.urgencyLevel,

required this.estimatedStock,

required this.maxPerCustomer,

this.shelfLocation,

required this.status,

required this.isFeatured,

this.erpTriggerRule,

});

factory FlashDeal.fromJson(Map<String, dynamic> json) =>

\_$FlashDealFromJson(json);

Map<String, dynamic> toJson() => \_$FlashDealToJson(this);

FlashDeal copyWith({

String? id,

String? productId,

String? storeId,

String? chainId,

String? productName,

String? brand,

String? storeName,

String? chainName,

int? originalPriceCents,

int? flashPriceCents,

int? discountPercentage,

DateTime? startsAt,

DateTime? expiresAt,

int? remainingSeconds,

String? urgencyLevel,

int? estimatedStock,

int? maxPerCustomer,

ShelfLocation? shelfLocation,

String? status,

bool? isFeatured,

String? erpTriggerRule,

}) {

return FlashDeal(

id: id ?? this.id,

productId: productId ?? this.productId,

storeId: storeId ?? this.storeId,

chainId: chainId ?? this.chainId,

productName: productName ?? this.productName,

brand: brand ?? this.brand,

storeName: storeName ?? this.storeName,

chainName: chainName ?? this.chainName,

originalPriceCents: originalPriceCents ?? this.originalPriceCents,

flashPriceCents: flashPriceCents ?? this.flashPriceCents,

discountPercentage: discountPercentage ?? this.discountPercentage,

startsAt: startsAt ?? this.startsAt,

expiresAt: expiresAt ?? this.expiresAt,

remainingSeconds: remainingSeconds ?? this.remainingSeconds,

urgencyLevel: urgencyLevel ?? this.urgencyLevel,

estimatedStock: estimatedStock ?? this.estimatedStock,

maxPerCustomer: maxPerCustomer ?? this.maxPerCustomer,

shelfLocation: shelfLocation ?? this.shelfLocation,

status: status ?? this.status,

isFeatured: isFeatured ?? this.isFeatured,

erpTriggerRule: erpTriggerRule ?? this.erpTriggerRule,

);

}

@override

List<Object?> get props => [

id, productId, storeId, chainId, productName, brand,

storeName, chainName, originalPriceCents, flashPriceCents,

discountPercentage, startsAt, expiresAt, remainingSeconds,

urgencyLevel, estimatedStock, maxPerCustomer, shelfLocation,

status, isFeatured, erpTriggerRule,

];

}

@JsonSerializable()

class ShelfLocation extends Equatable {

final String aisle;

final String shelf;

final int x;

final int y;

const ShelfLocation({

required this.aisle,

required this.shelf,

required this.x,

required this.y,

});

factory ShelfLocation.fromJson(Map<String, dynamic> json) =>

\_$ShelfLocationFromJson(json);

Map<String, dynamic> toJson() => \_$ShelfLocationToJson(this);

@override

List<Object> get props => [aisle, shelf, x, y];

}

**3.5 Storage Service - storage\_service.dart**

import 'dart:convert';

import 'package:shared\_preferences/shared\_preferences.dart';

class StorageService {

final SharedPreferences \_prefs;

StorageService(this.\_prefs);

// Generic storage methods

Future<bool> storeString(String key, String value) async {

return await \_prefs.setString(key, value);

}

String? getString(String key) {

return \_prefs.getString(key);

}

Future<bool> storeJson(String key, Map<String, dynamic> json) async {

final jsonString = jsonEncode(json);

return await \_prefs.setString(key, jsonString);

}

Map<String, dynamic>? getJson(String key) {

final jsonString = \_prefs.getString(key);

if (jsonString != null) {

return jsonDecode(jsonString) as Map<String, dynamic>;

}

return null;

}

Future<bool> storeList<T>(String key, List<T> items,

Map<String, dynamic> Function(T) toJson) async {

final jsonList = items.map((item) => toJson(item)).toList();

final jsonString = jsonEncode(jsonList);

return await \_prefs.setString(key, jsonString);

}

List<T> getList<T>(String key, T Function(Map<String, dynamic>) fromJson) {

final jsonString = \_prefs.getString(key);

if (jsonString != null) {

final jsonList = jsonDecode(jsonString) as List;

return jsonList

.map((json) => fromJson(json as Map<String, dynamic>))

.toList();

}

return [];

}

Future<bool> remove(String key) async {

return await \_prefs.remove(key);

}

Future<bool> clear() async {

return await \_prefs.clear();

}

// FlashFeed specific storage keys

static const String chainsKey = 'flashfeed\_chains';

static const String storesKey = 'flashfeed\_stores';

static const String productsKey = 'flashfeed\_products';

static const String o􀆯ersKey = 'flashfeed\_o􀆯ers';

static const String flashDealsKey = 'flashfeed\_flash\_deals';

static const String categoriesKey = 'flashfeed\_categories';

static const String userKey = 'flashfeed\_user';

static const String shoppingListsKey = 'flashfeed\_shopping\_lists';

static const String floorplansKey = 'flashfeed\_floorplans';

static const String lastDataUpdateKey = 'flashfeed\_last\_update';

}

**3.6 Mock Data Service - mock\_data\_service.dart**

import 'dart:async';

import 'dart:math';

import '../data/models/flash\_deal.dart';

import '../data/models/product.dart';

import '../data/models/store.dart';

import '../data/models/chain.dart';

import 'storage\_service.dart';

class MockDataService {

Timer? \_flashDealTimer;

Timer? \_countdownTimer;

final Random \_random = Random();

// Initialize all mock data on first app start

Future<void> initializeMockData(StorageService storage) async {

final lastUpdate = storage.getString(StorageService.lastDataUpdateKey);

if (lastUpdate == null) {

// First time initialization

await \_generateInitialMockData(storage);

await storage.storeString(StorageService.lastDataUpdateKey,

DateTime.now().toIso8601String());

}

// Start periodic updates

startPeriodicUpdates(storage);

}

void startPeriodicUpdates(StorageService storage) {

// Update flash deals every 2 hours

\_flashDealTimer?.cancel();

\_flashDealTimer = Timer.periodic(Duration(hours: 2), (timer) {

\_updateFlashDeals(storage);

});

// Update countdown timers every minute

\_countdownTimer?.cancel();

\_countdownTimer = Timer.periodic(Duration(minutes: 1), (timer) {

\_updateCountdownTimers(storage);

});

}

Future<void> \_generateInitialMockData(StorageService storage) async {

// Generate chains

final chains = \_generateMockChains();

await storage.storeList(StorageService.chainsKey, chains,

(chain) => chain.toJson());

// Generate stores

final stores = \_generateMockStores(chains);

await storage.storeList(StorageService.storesKey, stores,

(store) => store.toJson());

// Generate products

final products = \_generateMockProducts();

await storage.storeList(StorageService.productsKey, products,

(product) => product.toJson());

// Generate initial flash deals

final flashDeals = \_generateInitialFlashDeals(products, stores, chains);

await storage.storeList(StorageService.flashDealsKey, flashDeals,

(deal) => deal.toJson());

}

List<Chain> \_generateMockChains() {

return [

Chain(

id: 'edeka',

name: 'EDEKA',

displayName: 'EDEKA',

logoUrl: 'assets/images/logos/edeka.png',

primaryColor: '#005CA9',

isActive: true,

storeCount: 8,

),

Chain(

id: 'rewe',

name: 'REWE',

displayName: 'REWE',

logoUrl: 'assets/images/logos/rewe.png',

primaryColor: '#CC071E',

isActive: true,

storeCount: 8,

),

Chain(

id: 'aldi',

name: 'ALDI SÜD',

displayName: 'ALDI',

logoUrl: 'assets/images/logos/aldi.png',

primaryColor: '#00549F',

isActive: true,

storeCount: 6,

),

Chain(

id: 'lidl',

name: 'LIDL',

displayName: 'LIDL',

logoUrl: 'assets/images/logos/lidl.png',

primaryColor: '#0050AA',

isActive: true,

storeCount: 7,

),

Chain(

id: 'netto\_schwarz',

name: 'NETTO',

displayName: 'Netto',

logoUrl: 'assets/images/logos/netto-schwarz.png',

primaryColor: '#FFD100',

isActive: true,

storeCount: 5,

),

];

}

List<Store> \_generateMockStores(List<Chain> chains) {

final cities = [

{'name': 'Berlin', 'lat': 52.5200, 'lng': 13.4050},

{'name': 'Hamburg', 'lat': 53.5511, 'lng': 9.9937},

{'name': 'München', 'lat': 48.1351, 'lng': 11.5820},

{'name': 'Köln', 'lat': 50.9375, 'lng': 6.9603},

{'name': 'Frankfurt', 'lat': 50.1109, 'lng': 8.6821},

];

final storeTemplates = [

{'name': 'Hauptstraße', 'hasBeacon': true, 'size': 'large'},

{'name': 'Bahnhofstraße', 'hasBeacon': false, 'size': 'medium'},

{'name': 'Marktplatz', 'hasBeacon': true, 'size': 'small'},

{'name': 'Zentrum', 'hasBeacon': true, 'size': 'large'},

{'name': 'Nordstadt', 'hasBeacon': false, 'size': 'medium'},

{'name': 'Altstadt', 'hasBeacon': true, 'size': 'small'},

{'name': 'Alexanderplatz', 'hasBeacon': true, 'size': 'large'},

{'name': 'Potsdamer Platz', 'hasBeacon': true, 'size': 'medium'},

];

final stores = <Store>[];

int storeCounter = 1;

for (final chain in chains) {

for (int i = 0; i < chain.storeCount; i++) {

final city = cities[\_random.nextInt(cities.length)];

final template = storeTemplates[i % storeTemplates.length];

stores.add(Store(

id: 'store\_${storeCounter.toString().padLeft(3, '0')}',

chainId: chain.id,

name: '${chain.displayName} ${template['name']}',

street: '${template['name']} ${\_random.nextInt(200) + 1}',

zipCode: '${\_random.nextInt(90000) + 10000}',

city: city['name'] as String,

latitude: (city['lat'] as double) + (\_random.nextDouble() - 0.5) \* 0.1,

longitude: (city['lng'] as double) + (\_random.nextDouble() - 0.5) \* 0.1,

hasBeacon: template['hasBeacon'] as bool,

isActive: true,

size: template['size'] as String,

));

storeCounter++;

}

}

return stores;

}

List<Product> \_generateMockProducts() {

final productTemplates = {

'dairy': [

{'name': 'Vollmilch 3.5% 1L', 'brand': 'Landliebe', 'basePrice': 129},

{'name': 'Joghurt Natur 500g', 'brand': 'Danone', 'basePrice': 89},

{'name': 'Butter 250g', 'brand': 'Kerrygold', 'basePrice': 219},

],

'meat': [

{'name': 'Hähnchenbrust 1kg', 'brand': 'Wiesenhof', 'basePrice': 699},

{'name': 'Rinderhack 500g', 'brand': 'Meine Metzgerei', 'basePrice': 449},

],

'fruits': [

{'name': 'Äpfel 1kg', 'brand': 'Bio', 'basePrice': 249},

{'name': 'Bananen 1kg', 'brand': 'Chiquita', 'basePrice': 179},

],

'bakery': [

{'name': 'Vollkornbrot 500g', 'brand': 'Harry', 'basePrice': 189},

{'name': 'Brötchen 6 Stück', 'brand': 'Goldähren', 'basePrice': 149},

],

};

final products = <Product>[];

int productCounter = 1;

productTemplates.forEach((categoryId, templates) {

for (final template in templates) {

products.add(Product(

id: 'prod\_${productCounter.toString().padLeft(3, '0')}',

categoryId: categoryId,

name: template['name'] as String,

brand: template['brand'] as String,

basePriceCents: template['basePrice'] as int,

isActive: true,

));

productCounter++;

}

});

return products;

}

List<FlashDeal> \_generateInitialFlashDeals(

List<Product> products, List<Store> stores, List<Chain> chains) {

final flashDeals = <FlashDeal>[];

final currentTime = DateTime.now();

// Generate 15-20 initial flash deals

final dealCount = \_random.nextInt(6) + 15;

for (int i = 0; i < dealCount; i++) {

final product = products[\_random.nextInt(products.length)];

final store = stores.where((s) => s.hasBeacon).toList()

[\_random.nextInt(stores.where((s) => s.hasBeacon).length)];

final chain = chains.firstWhere((c) => c.id == store.chainId);

final discountPercent = \_random.nextInt(41) + 30; // 30-70%

final originalPrice = product.basePriceCents;

final flashPrice = (originalPrice \* (1 - discountPercent / 100)).round();

final durationHours = \_random.nextInt(6) + 1; // 1-6 hours

final expiresAt = currentTime.add(Duration(hours: durationHours));

final remainingHours = expiresAt.di􀆯erence(currentTime).inHours;

final urgencyLevel = remainingHours < 2 ? 'high' :

remainingHours < 4 ? 'medium' : 'low';

flashDeals.add(FlashDeal(

id: 'flash\_${(i + 1).toString().padLeft(3, '0')}',

productId: product.id,

storeId: store.id,

chainId: chain.id,

productName: product.name,

brand: product.brand,

storeName: store.name,

chainName: chain.name,

originalPriceCents: originalPrice,

flashPriceCents: flashPrice,

discountPercentage: discountPercent,

startsAt: currentTime,

expiresAt: expiresAt,

remainingSeconds: expiresAt.di􀆯erence(currentTime).inSeconds,

urgencyLevel: urgencyLevel,

estimatedStock: \_random.nextInt(50) + 5,

maxPerCustomer: \_random.nextInt(8) + 3,

shelfLocation: ShelfLocation(

aisle: '${String.fromCharCode(65 + \_random.nextInt(6))}${\_random.nextInt(8) + 1}',

shelf: \_random.nextBool() ? 'links' : 'rechts',

x: \_random.nextInt(600) + 100,

y: \_random.nextInt(400) + 100,

),

status: 'active',

isFeatured: \_random.nextDouble() > 0.8,

erpTriggerRule: discountPercent > 50 ? "MHD < 1 day" : "Stock > 30 items",

));

}

return flashDeals;

}

Future<void> \_updateFlashDeals(StorageService storage) async {

// Load current data

final products = storage.getList(StorageService.productsKey,

(json) => Product.fromJson(json));

final stores = storage.getList(StorageService.storesKey,

(json) => Store.fromJson(json));

final chains = storage.getList(StorageService.chainsKey,

(json) => Chain.fromJson(json));

final currentFlashDeals = storage.getList(StorageService.flashDealsKey,

(json) => FlashDeal.fromJson(json));

// Remove expired deals and add new ones

final now = DateTime.now();

final activeDeals = currentFlashDeals

.where((deal) => deal.expiresAt.isAfter(now))

.toList();

// Add new deals if below minimum

while (activeDeals.length < 15) {

final newDeal = \_generateSingleFlashDeal(products, stores, chains, now);

if (newDeal != null) activeDeals.add(newDeal);

}

// Save updated deals

await storage.storeList(StorageService.flashDealsKey, activeDeals,

(deal) => deal.toJson());

}

FlashDeal? \_generateSingleFlashDeal(List<Product> products, List<Store> stores,

List<Chain> chains, DateTime currentTime) {

if (products.isEmpty || stores.isEmpty || chains.isEmpty) return null;

final beaconStores = stores.where((s) => s.hasBeacon).toList();

if (beaconStores.isEmpty) return null;

final product = products[\_random.nextInt(products.length)];

final store = beaconStores[\_random.nextInt(beaconStores.length)];

final chain = chains.firstWhere((c) => c.id == store.chainId);

final discountPercent = \_random.nextInt(41) + 30;

final originalPrice = product.basePriceCents;

final flashPrice = (originalPrice \* (1 - discountPercent / 100)).round();

final durationHours = \_random.nextInt(6) + 1;

final expiresAt = currentTime.add(Duration(hours: durationHours));

final remainingHours = expiresAt.di􀆯erence(currentTime).inHours;

final urgencyLevel = remainingHours < 2 ? 'high' :

remainingHours < 4 ? 'medium' : 'low';

return FlashDeal(

id: 'flash\_${DateTime.now().millisecondsSinceEpoch}',

productId: product.id,

storeId: store.id,

chainId: chain.id,

productName: product.name,

brand: product.brand,

storeName: store.name,

chainName: chain.name,

originalPriceCents: originalPrice,

flashPriceCents: flashPrice,

discountPercentage: discountPercent,

startsAt: currentTime,

expiresAt: expiresAt,

remainingSeconds: expiresAt.di􀆯erence(currentTime).inSeconds,

urgencyLevel: urgencyLevel,

estimatedStock: \_random.nextInt(50) + 5,

maxPerCustomer: \_random.nextInt(8) + 3,

shelfLocation: ShelfLocation(

aisle: '${String.fromCharCode(65 + \_random.nextInt(6))}${\_random.nextInt(8) + 1}',

shelf: \_random.nextBool() ? 'links' : 'rechts',

x: \_random.nextInt(600) + 100,

y: \_random.nextInt(400) + 100,

),

status: 'active',

isFeatured: \_random.nextDouble() > 0.8,

erpTriggerRule: discountPercent > 50 ? "MHD < 1 day" : "Stock > 30 items",

);

}

Future<void> \_updateCountdownTimers(StorageService storage) async {

final flashDeals = storage.getList(StorageService.flashDealsKey,

(json) => FlashDeal.fromJson(json));

final now = DateTime.now();

final updatedDeals = flashDeals.map((deal) {

final remainingSeconds = deal.expiresAt.di􀆯erence(now).inSeconds;

final remainingHours = remainingSeconds / 3600;

final urgencyLevel = remainingHours < 1 ? 'high' :

remainingHours < 3 ? 'medium' : 'low';

return deal.copyWith(

remainingSeconds: math.max(0, remainingSeconds),

urgencyLevel: urgencyLevel,

status: remainingSeconds <= 0 ? 'expired' : deal.status,

);

}).where((deal) => deal.status == 'active').toList();

await storage.storeList(StorageService.flashDealsKey, updatedDeals,

(deal) => deal.toJson());

}

void dispose() {

\_flashDealTimer?.cancel();

\_countdownTimer?.cancel();

}

}

### BLoC & Widget Implementation

**1. BLoC Implementation Details**

**1.1 Flash Deals BLoC - flash\_deals\_bloc.dart**

import 'dart:async';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'package:equatable/equatable.dart';

import '../../data/models/flash\_deal.dart';

import '../../data/repositories/flash\_deal\_repository.dart';

import '../../core/services/mock\_data\_service.dart';

// Events

abstract class FlashDealsEvent extends Equatable {

const FlashDealsEvent();

@override

List<Object> get props => [];

}

class LoadFlashDealsEvent extends FlashDealsEvent {}

class RefreshFlashDealsEvent extends FlashDealsEvent {}

class FlashDealExpiredEvent extends FlashDealsEvent {

final String dealId;

const FlashDealExpiredEvent(this.dealId);

@override

List<Object> get props => [dealId];

}

class UpdateCountdownEvent extends FlashDealsEvent {}

// States

abstract class FlashDealsState extends Equatable {

const FlashDealsState();

@override

List<Object> get props => [];

}

class FlashDealsInitial extends FlashDealsState {}

class FlashDealsLoading extends FlashDealsState {}

class FlashDealsLoaded extends FlashDealsState {

final List<FlashDeal> flashDeals;

final List<FlashDeal> featuredDeals;

final List<FlashDeal> expiringDeals;

const FlashDealsLoaded({

required this.flashDeals,

required this.featuredDeals,

required this.expiringDeals,

});

@override

List<Object> get props => [flashDeals, featuredDeals, expiringDeals];

}

class FlashDealsError extends FlashDealsState {

final String message;

const FlashDealsError(this.message);

@override

List<Object> get props => [message];

}

// BLoC

class FlashDealsBloc extends Bloc<FlashDealsEvent, FlashDealsState> {

final FlashDealRepository flashDealRepository;

final MockDataService mockDataService;

Timer? \_countdownTimer;

FlashDealsBloc({

required this.flashDealRepository,

required this.mockDataService,

}) : super(FlashDealsInitial()) {

on<LoadFlashDealsEvent>(\_onLoadFlashDeals);

on<RefreshFlashDealsEvent>(\_onRefreshFlashDeals);

on<FlashDealExpiredEvent>(\_onFlashDealExpired);

on<UpdateCountdownEvent>(\_onUpdateCountdown);

// Start countdown timer

\_startCountdownTimer();

}

void \_startCountdownTimer() {

\_countdownTimer?.cancel();

\_countdownTimer = Timer.periodic(

const Duration(seconds: 1),

(timer) => add(UpdateCountdownEvent()),

);

}

@override

Future<void> close() {

\_countdownTimer?.cancel();

return super.close();

}

Future<void> \_onLoadFlashDeals(

LoadFlashDealsEvent event,

Emitter<FlashDealsState> emit,

) async {

emit(FlashDealsLoading());

try {

final flashDeals = await flashDealRepository.getActiveFlashDeals();

emit(\_categorizeDeals(flashDeals));

} catch (e) {

emit(FlashDealsError('Failed to load flash deals: $e'));

}

}

Future<void> \_onRefreshFlashDeals(

RefreshFlashDealsEvent event,

Emitter<FlashDealsState> emit,

) async {

try {

await flashDealRepository.refreshFlashDeals();

final flashDeals = await flashDealRepository.getActiveFlashDeals();

emit(\_categorizeDeals(flashDeals));

} catch (e) {

emit(FlashDealsError('Failed to refresh flash deals: $e'));

}

}

Future<void> \_onFlashDealExpired(

FlashDealExpiredEvent event,

Emitter<FlashDealsState> emit,

) async {

if (state is FlashDealsLoaded) {

final currentState = state as FlashDealsLoaded;

final updatedDeals = currentState.flashDeals

.where((deal) => deal.id != event.dealId)

.toList();

emit(\_categorizeDeals(updatedDeals));

}

}

Future<void> \_onUpdateCountdown(

UpdateCountdownEvent event,

Emitter<FlashDealsState> emit,

) async {

if (state is FlashDealsLoaded) {

final currentState = state as FlashDealsLoaded;

final now = DateTime.now();

final updatedDeals = currentState.flashDeals.map((deal) {

final remainingSeconds = deal.expiresAt.di􀆯erence(now).inSeconds;

if (remainingSeconds <= 0) {

add(FlashDealExpiredEvent(deal.id));

return null;

}

final remainingHours = remainingSeconds / 3600;

final urgencyLevel = remainingHours < 1 ? 'high' :

remainingHours < 3 ? 'medium' : 'low';

return deal.copyWith(

remainingSeconds: remainingSeconds,

urgencyLevel: urgencyLevel,

);

}).where((deal) => deal != null).cast<FlashDeal>().toList();

if (updatedDeals.length != currentState.flashDeals.length ||

\_dealsChanged(updatedDeals, currentState.flashDeals)) {

emit(\_categorizeDeals(updatedDeals));

}

}

}

bool \_dealsChanged(List<FlashDeal> newDeals, List<FlashDeal> oldDeals) {

if (newDeals.length != oldDeals.length) return true;

for (int i = 0; i < newDeals.length; i++) {

if (newDeals[i].remainingSeconds != oldDeals[i].remainingSeconds ||

newDeals[i].urgencyLevel != oldDeals[i].urgencyLevel) {

return true;

}

}

return false;

}

FlashDealsLoaded \_categorizeDeals(List<FlashDeal> deals) {

final featuredDeals = deals.where((deal) => deal.isFeatured).toList();

final expiringDeals = deals

.where((deal) => deal.remainingSeconds < 1800) // < 30 minutes

.toList();

return FlashDealsLoaded(

flashDeals: deals,

featuredDeals: featuredDeals,

expiringDeals: expiringDeals,

);

}

}

**1.2 Main Navigation BLoC - main\_navigation\_bloc.dart**

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'package:equatable/equatable.dart';

// Events

abstract class MainNavigationEvent extends Equatable {

const MainNavigationEvent();

@override

List<Object> get props => [];

}

class TabChangedEvent extends MainNavigationEvent {

final int tabIndex;

const TabChangedEvent(this.tabIndex);

@override

List<Object> get props => [tabIndex];

}

class ToggleSettingsEvent extends MainNavigationEvent {}

class CloseSettingsEvent extends MainNavigationEvent {}

// States

class MainNavigationState extends Equatable {

final int currentTabIndex;

final bool isSettingsOpen;

const MainNavigationState({

this.currentTabIndex = 0,

this.isSettingsOpen = false,

});

MainNavigationState copyWith({

int? currentTabIndex,

bool? isSettingsOpen,

}) {

return MainNavigationState(

currentTabIndex: currentTabIndex ?? this.currentTabIndex,

isSettingsOpen: isSettingsOpen ?? this.isSettingsOpen,

);

}

@override

List<Object> get props => [currentTabIndex, isSettingsOpen];

}

// BLoC

class MainNavigationBloc extends Bloc<MainNavigationEvent, MainNavigationState> {

MainNavigationBloc() : super(const MainNavigationState()) {

on<TabChangedEvent>(\_onTabChanged);

on<ToggleSettingsEvent>(\_onToggleSettings);

on<CloseSettingsEvent>(\_onCloseSettings);

}

void \_onTabChanged(

TabChangedEvent event,

Emitter<MainNavigationState> emit,

) {

emit(state.copyWith(

currentTabIndex: event.tabIndex,

isSettingsOpen: false, // Close settings when switching tabs

));

}

void \_onToggleSettings(

ToggleSettingsEvent event,

Emitter<MainNavigationState> emit,

) {

emit(state.copyWith(isSettingsOpen: !state.isSettingsOpen));

}

void \_onCloseSettings(

CloseSettingsEvent event,

Emitter<MainNavigationState> emit,

) {

emit(state.copyWith(isSettingsOpen: false));

}

}

**2. Widget Implementation Details**

**2.1 Main Screen - main\_screen.dart**

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import '../blocs/main\_navigation/main\_navigation\_bloc.dart';

import '../widgets/common/app\_header.dart';

import '../widgets/common/tab\_navigation.dart';

import 'o􀆯ers\_screen.dart';

import 'map\_screen.dart';

import 'flash\_deals\_screen.dart';

import 'settings\_screen.dart';

import '../../core/constants/app\_colors.dart';

class MainScreen extends StatelessWidget {

const MainScreen({super.key});

@override

Widget build(BuildContext context) {

return BlocBuilder<MainNavigationBloc, MainNavigationState>(

builder: (context, state) {

return Sca􀆯old(

backgroundColor: AppColors.backgroundLight,

body: Stack(

children: [

Column(

children: [

const AppHeader(),

Expanded(

child: IndexedStack(

index: state.currentTabIndex,

children: const [

O􀆯ersScreen(),

MapScreen(),

FlashDealsScreen(),

],

),

),

const TabNavigation(),

],

),

// Settings overlay

if (state.isSettingsOpen)

Container(

color: Colors.black.withOpacity(0.5),

child: const SettingsScreen(),

),

],

),

);

},

);

}

}

**2.2 App Header Widget - app\_header.dart**

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'package:lucide\_icons/lucide\_icons.dart';

import '../../blocs/main\_navigation/main\_navigation\_bloc.dart';

import '../../core/constants/app\_colors.dart';

import '../../core/constants/app\_text\_styles.dart';

import '../../core/constants/app\_dimensions.dart';

class AppHeader extends StatelessWidget {

const AppHeader({super.key});

@override

Widget build(BuildContext context) {

return Container(

height: AppDimensions.headerHeight,

width: double.infinity,

decoration: const BoxDecoration(

color: AppColors.primaryGreen,

boxShadow: [

BoxShadow(

color: Colors.black12,

blurRadius: 4,

o􀆯set: O􀆯set(0, 2),

),

],

),

child: SafeArea(

child: Padding(

padding: const EdgeInsets.symmetric(

horizontal: AppDimensions.paddingMedium,

),

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

// App Logo and Name

Row(

children: [

Container(

width: 32,

height: 32,

decoration: const BoxDecoration(

color: AppColors.textWhite,

shape: BoxShape.circle,

),

child: const Icon(

LucideIcons.zap,

color: AppColors.primaryGreen,

size: 20,

),

),

const SizedBox(width: AppDimensions.spacingSmall),

Text(

'FlashFeed',

style: AppTextStyles.headerTitle.copyWith(

color: AppColors.textWhite,

),

),

],

),

// Hamburger Menu

GestureDetector(

onTap: () {

context.read<MainNavigationBloc>().add(ToggleSettingsEvent());

},

child: Container(

width: 44,

height: 44,

decoration: BoxDecoration(

color: Colors.white.withOpacity(0.1),

borderRadius: BorderRadius.circular(8),

),

child: const Icon(

LucideIcons.menu,

color: AppColors.textWhite,

size: 24,

),

),

),

],

),

),

),

);

}

}

**2.3 Tab Navigation - tab\_navigation.dart**

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'package:lucide\_icons/lucide\_icons.dart';

import '../../blocs/main\_navigation/main\_navigation\_bloc.dart';

import '../../core/constants/app\_colors.dart';

import '../../core/constants/app\_text\_styles.dart';

import '../../core/constants/app\_dimensions.dart';

class TabNavigation extends StatelessWidget {

const TabNavigation({super.key});

@override

Widget build(BuildContext context) {

return BlocBuilder<MainNavigationBloc, MainNavigationState>(

builder: (context, state) {

return Container(

height: AppDimensions.tabBarHeight,

decoration: const BoxDecoration(

color: AppColors.backgroundLight,

border: Border(

top: BorderSide(

color: AppColors.secondaryGray,

width: 1,

),

),

),

child: Row(

children: [

\_TabItem(

icon: LucideIcons.shoppingCart,

label: 'Angebote',

isActive: state.currentTabIndex == 0,

onTap: () => \_onTabTapped(context, 0),

),

\_TabItem(

icon: LucideIcons.mapPin,

label: 'Karte',

isActive: state.currentTabIndex == 1,

onTap: () => \_onTabTapped(context, 1),

),

\_TabItem(

icon: LucideIcons.zap,

label: 'Flash',

isActive: state.currentTabIndex == 2,

onTap: () => \_onTabTapped(context, 2),

),

],

),

);

},

);

}

void \_onTabTapped(BuildContext context, int index) {

context.read<MainNavigationBloc>().add(TabChangedEvent(index));

}

}

class \_TabItem extends StatelessWidget {

final IconData icon;

final String label;

final bool isActive;

final VoidCallback onTap;

const \_TabItem({

required this.icon,

required this.label,

required this.isActive,

required this.onTap,

});

@override

Widget build(BuildContext context) {

return Expanded(

child: GestureDetector(

onTap: onTap,

child: Container(

decoration: BoxDecoration(

color: isActive ? AppColors.primaryGreen : Colors.transparent,

borderRadius: BorderRadius.circular(8),

),

margin: const EdgeInsets.symmetric(

horizontal: AppDimensions.spacingSmall,

vertical: AppDimensions.spacingSmall,

),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Icon(

icon,

size: 24,

color: isActive ? AppColors.textWhite : AppColors.secondaryGray,

),

const SizedBox(height: 4),

Text(

label,

style: AppTextStyles.tabLabel.copyWith(

color: isActive ? AppColors.textWhite : AppColors.textSecondary,

),

),

],

),

),

),

);

}

}

**2.4 Flash Deal Card - flash\_deal\_card.dart**

import 'package:flutter/material.dart';

import 'package:lucide\_icons/lucide\_icons.dart';

import '../../../data/models/flash\_deal.dart';

import '../../core/constants/app\_colors.dart';

import '../../core/constants/app\_text\_styles.dart';

import '../../core/constants/app\_dimensions.dart';

import '../../core/utils/price\_utils.dart';

import '../common/countdown\_timer.dart';

import 'urgency\_badge.dart';

class FlashDealCard extends StatelessWidget {

final FlashDeal flashDeal;

final VoidCallback? onTap;

const FlashDealCard({

super.key,

required this.flashDeal,

this.onTap,

});

@override

Widget build(BuildContext context) {

return GestureDetector(

onTap: onTap,

child: Container(

margin: const EdgeInsets.only(bottom: AppDimensions.spacingMedium),

decoration: BoxDecoration(

color: Colors.white,

borderRadius: BorderRadius.circular(12),

border: const Border(

left: BorderSide(color: AppColors.primaryRed, width: 4),

),

boxShadow: [

BoxShadow(

color: Colors.black.withOpacity(0.1),

blurRadius: 8,

o􀆯set: const O􀆯set(0, 3),

),

],

),

child: Padding(

padding: const EdgeInsets.all(AppDimensions.paddingMedium),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

// Header Row

Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

CountdownTimer(

remainingSeconds: flashDeal.remainingSeconds,

urgencyLevel: flashDeal.urgencyLevel,

),

Row(

children: [

UrgencyBadge(urgencyLevel: flashDeal.urgencyLevel),

const SizedBox(width: AppDimensions.spacingSmall),

\_DiscountBadge(discount: flashDeal.discountPercentage),

],

),

],

),

const SizedBox(height: AppDimensions.spacingMedium),

// Product Info

Row(

children: [

Expanded(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

flashDeal.productName,

style: AppTextStyles.productName,

maxLines: 2,

overflow: TextOverflow.ellipsis,

),

if (flashDeal.brand.isNotEmpty) ...[

const SizedBox(height: AppDimensions.spacingXSmall),

Text(

flashDeal.brand,

style: AppTextStyles.productBrand,

),

],

],

),

),

const SizedBox(width: AppDimensions.spacingMedium),

Column(

crossAxisAlignment: CrossAxisAlignment.end,

children: [

Text(

PriceUtils.formatPrice(flashDeal.originalPriceCents),

style: AppTextStyles.originalPrice,

),

Text(

PriceUtils.formatPrice(flashDeal.flashPriceCents),

style: AppTextStyles.discountedPrice,

),

],

),

],

),

const SizedBox(height: AppDimensions.spacingMedium),

// Store Info

Row(

children: [

Icon(

LucideIcons.mapPin,

size: 16,

color: AppColors.textSecondary,

),

const SizedBox(width: AppDimensions.spacingXSmall),

Expanded(

child: Text(

'${flashDeal.storeName} • ${flashDeal.chainName}',

style: AppTextStyles.storeInfo,

overflow: TextOverflow.ellipsis,

),

),

],

),

// Stock Info

if (flashDeal.estimatedStock < 20) ...[

const SizedBox(height: AppDimensions.spacingSmall),

Row(

children: [

Icon(

LucideIcons.package,

size: 16,

color: flashDeal.estimatedStock < 10

? AppColors.warning

: AppColors.textSecondary,

),

const SizedBox(width: AppDimensions.spacingXSmall),

Text(

'Noch ${flashDeal.estimatedStock} verfügbar',

style: AppTextStyles.stockInfo.copyWith(

color: flashDeal.estimatedStock < 10

? AppColors.warning

: AppColors.textSecondary,

),

),

],

),

],

const SizedBox(height: AppDimensions.spacingMedium),

// Action Button

SizedBox(

width: double.infinity,

child: ElevatedButton(

onPressed: onTap,

style: ElevatedButton.styleFrom(

backgroundColor: AppColors.primaryGreen,

foregroundColor: AppColors.textWhite,

padding: const EdgeInsets.symmetric(

vertical: AppDimensions.paddingMedium,

),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(8),

),

),

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Icon(

LucideIcons.navigation,

size: 20,

),

const SizedBox(width: AppDimensions.spacingSmall),

const Text('Zum Lageplan'),

],

),

),

),

],

),

),

),

);

}

}

class \_DiscountBadge extends StatelessWidget {

final int discount;

const \_DiscountBadge({required this.discount});

@override

Widget build(BuildContext context) {

return Container(

padding: const EdgeInsets.symmetric(

horizontal: AppDimensions.paddingSmall,

vertical: 4,

),

decoration: BoxDecoration(

color: AppColors.primaryRed,

borderRadius: BorderRadius.circular(20),

),

child: Text(

'-$discount%',

style: AppTextStyles.discountBadge,

),

);

}

}

**2.5 Countdown Timer Widget - countdown\_timer.dart**

import 'package:flutter/material.dart';

import '../../core/constants/app\_colors.dart';

import '../../core/constants/app\_text\_styles.dart';

import '../../core/utils/date\_utils.dart';

class CountdownTimer extends StatefulWidget {

final int remainingSeconds;

final String urgencyLevel;

const CountdownTimer({

super.key,

required this.remainingSeconds,

required this.urgencyLevel,

});

@override

State<CountdownTimer> createState() => \_CountdownTimerState();

}

class \_CountdownTimerState extends State<CountdownTimer>

with SingleTickerProviderStateMixin {

late AnimationController \_blinkController;

late Animation<double> \_blinkAnimation;

@override

void initState() {

super.initState();

\_blinkController = AnimationController(

duration: const Duration(milliseconds: 500),

vsync: this,

);

\_blinkAnimation = Tween<double>(begin: 1.0, end: 0.3).animate(

CurvedAnimation(parent: \_blinkController, curve: Curves.easeInOut),

);

if (widget.urgencyLevel == 'high') {

\_blinkController.repeat(reverse: true);

}

}

@override

void didUpdateWidget(CountdownTimer oldWidget) {

super.didUpdateWidget(oldWidget);

if (widget.urgencyLevel == 'high' && !\_blinkController.isAnimating) {

\_blinkController.repeat(reverse: true);

} else if (widget.urgencyLevel != 'high' && \_blinkController.isAnimating) {

\_blinkController.stop();

\_blinkController.reset();

}

}

@override

void dispose() {

\_blinkController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

final timeString = DateUtils.formatCountdown(widget.remainingSeconds);

final color = \_getTimerColor(widget.urgencyLevel);

if (widget.urgencyLevel == 'high') {

return AnimatedBuilder(

animation: \_blinkAnimation,

builder: (context, child) {

return Opacity(

opacity: \_blinkAnimation.value,

child: \_buildTimer(timeString, color),

);

},

);

}

return \_buildTimer(timeString, color);

}

Widget \_buildTimer(String timeString, Color color) {

return Container(

padding: const EdgeInsets.symmetric(horizontal: 12, vertical: 6),

decoration: BoxDecoration(

color: color.withOpacity(0.1),

borderRadius: BorderRadius.circular(20),

border: Border.all(color: color, width: 1),

),

child: Row(

mainAxisSize: MainAxisSize.min,

children: [

Icon(

Icons.timer\_outlined,

size: 16,

color: color,

),

const SizedBox(width: 4),

Text(

timeString,

style: AppTextStyles.countdownTimer.copyWith(color: color),

),

],

),

);

}

Color \_getTimerColor(String urgencyLevel) {

switch (urgencyLevel) {

case 'high':

return AppColors.primaryRed;

case 'medium':

return AppColors.secondaryOrange;

case 'low':

default:

return AppColors.primaryGreen;

}

}

}

**2.6 Chain Selector Widget - chain\_selector.dart**

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import '../../../data/models/chain.dart';

import '../../../data/models/user.dart';

import '../../blocs/o􀆯ers/o􀆯ers\_bloc.dart';

import '../../core/constants/app\_colors.dart';

import '../../core/constants/app\_dimensions.dart';

import '../../core/constants/app\_text\_styles.dart';

class ChainSelector extends StatelessWidget {

final List<Chain> chains;

final List<String> selectedChainIds;

final User user;

const ChainSelector({

super.key,

required this.chains,

required this.selectedChainIds,

required this.user,

});

@override

Widget build(BuildContext context) {

return Container(

height: 80,

padding: const EdgeInsets.symmetric(

horizontal: AppDimensions.paddingMedium,

vertical: AppDimensions.paddingSmall,

),

decoration: const BoxDecoration(

color: AppColors.backgroundLight,

border: Border(

bottom: BorderSide(

color: AppColors.secondaryGray,

width: 0.5,

),

),

),

child: ListView.builder(

scrollDirection: Axis.horizontal,

itemCount: chains.length,

itemBuilder: (context, index) {

final chain = chains[index];

final isSelected = selectedChainIds.contains(chain.id);

final isAvailable = \_isChainAvailable(chain.id, user);

return Padding(

padding: EdgeInsets.only(

right: index < chains.length - 1 ? AppDimensions.spacingMedium : 0,

),

child: ChainIcon(

chain: chain,

isSelected: isSelected,

isAvailable: isAvailable,

onTap: isAvailable

? () => \_onChainTapped(context, chain.id)

: null,

),

);

},

),

);

}

bool \_isChainAvailable(String chainId, User user) {

// Basic plan: only 1 chain allowed

if (user.subscriptionPlan == 'basic') {

return selectedChainIds.isEmpty || selectedChainIds.contains(chainId);

}

// Premium/Family: all chains available

return true;

}

void \_onChainTapped(BuildContext context, String chainId) {

context.read<O􀆯ersBloc>().add(ToggleChainEvent(chainId));

}

}

class ChainIcon extends StatelessWidget {

final Chain chain;

final bool isSelected;

final bool isAvailable;

final VoidCallback? onTap;

const ChainIcon({

super.key,

required this.chain,

required this.isSelected,

required this.isAvailable,

this.onTap,

});

@override

Widget build(BuildContext context) {

return GestureDetector(

onTap: onTap,

child: Column(

children: [

Container(

width: 60,

height: 60,

decoration: BoxDecoration(

borderRadius: BorderRadius.circular(8),

border: Border.all(

color: isSelected

? AppColors.primaryGreen

: AppColors.secondaryGray,

width: isSelected ? 2 : 1,

),

boxShadow: isSelected ? [

BoxShadow(

color: AppColors.primaryGreen.withOpacity(0.3),

blurRadius: 8,

o􀆯set: const O􀆯set(0, 2),

),

] : null,

),

child: ClipRRect(

borderRadius: BorderRadius.circular(6),

child: Stack(

children: [

Image.asset(

chain.logoUrl,

width: 60,

height: 60,

fit: BoxFit.cover,

),

if (!isAvailable)

Container(

width: 60,

height: 60,

decoration: BoxDecoration(

color: Colors.black.withOpacity(0.6),

borderRadius: BorderRadius.circular(6),

),

child: const Icon(

Icons.lock\_outline,

color: Colors.white,

size: 24,

),

),

],

),

),

),

const SizedBox(height: 4),

Text(

chain.displayName,

style: AppTextStyles.chainLabel.copyWith(

color: isSelected

? AppColors.primaryGreen

: isAvailable

? AppColors.textPrimary

: AppColors.textSecondary,

fontWeight: isSelected ? FontWeight.w600 : FontWeight.normal,

),

),

],

),

);

}

}

**3. Utility Classes**

**3.1 Price Utils - price\_utils.dart**

class PriceUtils {

static String formatPrice(int priceCents) {

final euros = priceCents / 100;

return '${euros.toStringAsFixed(2).replaceAll('.', ',')} €';

}

static String formatPriceCompact(int priceCents) {

final euros = priceCents / 100;

if (euros == euros.toInt()) {

return '${euros.toInt()} €';

}

return '${euros.toStringAsFixed(2).replaceAll('.', ',')} €';

}

static String calculateSavings(int originalPrice, int discountedPrice) {

final savings = originalPrice - discountedPrice;

return formatPrice(savings);

}

static int calculateDiscountPercentage(int originalPrice, int discountedPrice) {

if (originalPrice == 0) return 0;

return (((originalPrice - discountedPrice) / originalPrice) \* 100).round();

}

}

**3.2 Date Utils - date\_utils.dart**

class DateUtils {

static String formatCountdown(int remainingSeconds) {

if (remainingSeconds <= 0) return '00:00:00';

final hours = remainingSeconds ~/ 3600;

final minutes = (remainingSeconds % 3600) ~/ 60;

final seconds = remainingSeconds % 60;

return '${hours.toString().padLeft(2, '0')}:'

'${minutes.toString().padLeft(2, '0')}:'

'${seconds.toString().padLeft(2, '0')}';

}

static String formatTimeRemaining(int remainingSeconds) {

if (remainingSeconds <= 0) return 'Abgelaufen';

final hours = remainingSeconds ~/ 3600;

final minutes = (remainingSeconds % 3600) ~/ 60;

if (hours > 0) {

return 'noch ${hours}h ${minutes}min';

} else if (minutes > 0) {

return 'noch ${minutes}min';

} else {

return 'noch ${remainingSeconds}s';

}

}

static String formatRelativeTime(DateTime dateTime) {

final now = DateTime.now();

final di􀆯erence = now.di􀆯erence(dateTime);

if (di􀆯erence.inMinutes < 1) {

return 'vor wenigen Sekunden';

} else if (di􀆯erence.inMinutes < 60) {

return 'vor ${di􀆯erence.inMinutes} Minuten';

} else if (di􀆯erence.inHours < 24) {

return 'vor ${di􀆯erence.inHours} Stunden';

} else {

return 'vor ${di􀆯erence.inDays} Tagen';

}

}

}

### Repository & Deployment Implementation

**1. Repository Implementation (Data Layer)**

**1.1 Flash Deal Repository - flash\_deal\_repository.dart**

import '../models/flash\_deal.dart';

import '../datasources/local\_storage\_datasource.dart';

import '../../core/services/storage\_service.dart';

class FlashDealRepository {

final StorageService storageService;

late final LocalStorageDataSource \_localDataSource;

FlashDealRepository(this.storageService) {

\_localDataSource = LocalStorageDataSource(storageService);

}

Future<List<FlashDeal>> getActiveFlashDeals() async {

try {

final deals = await \_localDataSource.getFlashDeals();

final now = DateTime.now();

// Filter nur aktive und nicht-abgelaufene Deals

return deals

.where((deal) =>

deal.status == 'active' &&

deal.expiresAt.isAfter(now))

.toList();

} catch (e) {

throw Exception('Failed to load flash deals: $e');

}

}

Future<List<FlashDeal>> getFlashDealsByStore(String storeId) async {

try {

final deals = await getActiveFlashDeals();

return deals.where((deal) => deal.storeId == storeId).toList();

} catch (e) {

throw Exception('Failed to load flash deals for store: $e');

}

}

Future<List<FlashDeal>> getFlashDealsByChain(String chainId) async {

try {

final deals = await getActiveFlashDeals();

return deals.where((deal) => deal.chainId == chainId).toList();

} catch (e) {

throw Exception('Failed to load flash deals for chain: $e');

}

}

Future<List<FlashDeal>> getFeaturedFlashDeals() async {

try {

final deals = await getActiveFlashDeals();

return deals.where((deal) => deal.isFeatured).toList();

} catch (e) {

throw Exception('Failed to load featured flash deals: $e');

}

}

Future<List<FlashDeal>> getExpiringFlashDeals({int minutesThreshold = 30}) async {

try {

final deals = await getActiveFlashDeals();

final threshold = DateTime.now().add(Duration(minutes: minutesThreshold));

return deals

.where((deal) => deal.expiresAt.isBefore(threshold))

.toList();

} catch (e) {

throw Exception('Failed to load expiring flash deals: $e');

}

}

Future<FlashDeal?> getFlashDealById(String id) async {

try {

final deals = await \_localDataSource.getFlashDeals();

return deals.firstWhere(

(deal) => deal.id == id,

orElse: () => throw Exception('Flash deal not found'),

);

} catch (e) {

return null;

}

}

Future<void> refreshFlashDeals() async {

try {

// Trigger refresh über MockDataService

// In einer echten App würde hier ein API-Call gemacht

await \_localDataSource.refreshData();

} catch (e) {

throw Exception('Failed to refresh flash deals: $e');

}

}

Future<void> markFlashDealAsViewed(String dealId) async {

try {

final deals = await \_localDataSource.getFlashDeals();

final dealIndex = deals.indexWhere((deal) => deal.id == dealId);

if (dealIndex != -1) {

// In einer echten App würde hier Analytics-Tracking stattfinden

// Hier können wir Views in LocalStorage zählen

await \_recordActivity('flash\_deal\_viewed', dealId);

}

} catch (e) {

// Silent fail für Analytics

}

}

Future<void> markFlashDealAsClicked(String dealId) async {

try {

await \_recordActivity('flash\_deal\_clicked', dealId);

} catch (e) {

// Silent fail für Analytics

}

}

Future<void> \_recordActivity(String activityType, String dealId) async {

try {

final activities = storageService.getList('user\_activities',

(json) => UserActivity.fromJson(json)) ?? [];

activities.add(UserActivity(

id: 'activity\_${DateTime.now().millisecondsSinceEpoch}',

activityType: activityType,

entityId: dealId,

entityType: 'flash\_deal',

timestamp: DateTime.now(),

));

// Nur letzten 100 Aktivitäten behalten

final recentActivities = activities.length > 100

? activities.skip(activities.length - 100).toList()

: activities;

await storageService.storeList('user\_activities', recentActivities,

(activity) => activity.toJson());

} catch (e) {

// Silent fail

}

}

}

**1.2 Local Storage Datasource - local\_storage\_datasource.dart**

import '../models/chain.dart';

import '../models/store.dart';

import '../models/product.dart';

import '../models/flash\_deal.dart';

import '../models/o􀆯er.dart';

import '../models/category.dart';

import '../models/user.dart';

import '../models/floorplan.dart';

import '../../core/services/storage\_service.dart';

import 'mock\_data\_generator.dart';

class LocalStorageDataSource {

final StorageService storageService;

final MockDataGenerator mockDataGenerator = MockDataGenerator();

LocalStorageDataSource(this.storageService);

// Chains

Future<List<Chain>> getChains() async {

final chains = storageService.getList(StorageService.chainsKey,

(json) => Chain.fromJson(json));

if (chains.isEmpty) {

final mockChains = mockDataGenerator.generateChains();

await storeChains(mockChains);

return mockChains;

}

return chains;

}

Future<void> storeChains(List<Chain> chains) async {

await storageService.storeList(StorageService.chainsKey, chains,

(chain) => chain.toJson());

}

// Stores

Future<List<Store>> getStores() async {

final stores = storageService.getList(StorageService.storesKey,

(json) => Store.fromJson(json));

if (stores.isEmpty) {

final chains = await getChains();

final mockStores = mockDataGenerator.generateStores(chains);

await storeStores(mockStores);

return mockStores;

}

return stores;

}

Future<void> storeStores(List<Store> stores) async {

await storageService.storeList(StorageService.storesKey, stores,

(store) => store.toJson());

}

Future<List<Store>> getStoresByChain(String chainId) async {

final stores = await getStores();

return stores.where((store) => store.chainId == chainId).toList();

}

Future<List<Store>> getNearbyStores(double lat, double lng, double radiusKm) async {

final stores = await getStores();

return stores.where((store) {

final distance = \_calculateDistance(lat, lng, store.latitude, store.longitude);

return distance <= radiusKm;

}).toList();

}

// Products

Future<List<Product>> getProducts() async {

final products = storageService.getList(StorageService.productsKey,

(json) => Product.fromJson(json));

if (products.isEmpty) {

final mockProducts = mockDataGenerator.generateProducts();

await storeProducts(mockProducts);

return mockProducts;

}

return products;

}

Future<void> storeProducts(List<Product> products) async {

await storageService.storeList(StorageService.productsKey, products,

(product) => product.toJson());

}

// Flash Deals

Future<List<FlashDeal>> getFlashDeals() async {

final flashDeals = storageService.getList(StorageService.flashDealsKey,

(json) => FlashDeal.fromJson(json));

if (flashDeals.isEmpty) {

await \_generateInitialFlashDeals();

return storageService.getList(StorageService.flashDealsKey,

(json) => FlashDeal.fromJson(json));

}

return flashDeals;

}

Future<void> storeFlashDeals(List<FlashDeal> flashDeals) async {

await storageService.storeList(StorageService.flashDealsKey, flashDeals,

(deal) => deal.toJson());

}

Future<void> \_generateInitialFlashDeals() async {

final products = await getProducts();

final stores = await getStores();

final chains = await getChains();

final flashDeals = mockDataGenerator.generateFlashDeals(

products, stores, chains);

await storeFlashDeals(flashDeals);

}

// Categories

Future<List<Category>> getCategories() async {

final categories = storageService.getList(StorageService.categoriesKey,

(json) => Category.fromJson(json));

if (categories.isEmpty) {

final mockCategories = mockDataGenerator.generateCategories();

await storeCategories(mockCategories);

return mockCategories;

}

return categories;

}

Future<void> storeCategories(List<Category> categories) async {

await storageService.storeList(StorageService.categoriesKey, categories,

(category) => category.toJson());

}

// Floorplans

Future<Floorplan?> getFloorplanByStore(String storeId) async {

final floorplans = storageService.getList(StorageService.floorplansKey,

(json) => Floorplan.fromJson(json));

try {

return floorplans.firstWhere((fp) => fp.storeId == storeId);

} catch (e) {

// Generate floorplan if not exists

final store = await \_getStoreById(storeId);

if (store != null && store.hasBeacon) {

final floorplan = mockDataGenerator.generateFloorplan(store);

await \_addFloorplan(floorplan);

return floorplan;

}

return null;

}

}

Future<void> \_addFloorplan(Floorplan floorplan) async {

final floorplans = storageService.getList(StorageService.floorplansKey,

(json) => Floorplan.fromJson(json));

floorplans.add(floorplan);

await storageService.storeList(StorageService.floorplansKey, floorplans,

(fp) => fp.toJson());

}

Future<Store?> \_getStoreById(String storeId) async {

final stores = await getStores();

try {

return stores.firstWhere((store) => store.id == storeId);

} catch (e) {

return null;

}

}

// User Data

Future<User?> getCurrentUser() async {

final userData = storageService.getJson(StorageService.userKey);

if (userData != null) {

return User.fromJson(userData);

}

// Create default user if none exists

final defaultUser = mockDataGenerator.generateDefaultUser();

await storeUser(defaultUser);

return defaultUser;

}

Future<void> storeUser(User user) async {

await storageService.storeJson(StorageService.userKey, user.toJson());

}

// Utility Methods

Future<void> refreshData() async {

// Simuliere Daten-Refresh durch Regenerierung

final chains = await getChains();

final stores = await getStores();

final products = await getProducts();

// Generiere neue Flash Deals

final newFlashDeals = mockDataGenerator.generateFlashDeals(

products, stores, chains);

await storeFlashDeals(newFlashDeals);

// Update timestamp

await storageService.storeString(StorageService.lastDataUpdateKey,

DateTime.now().toIso8601String());

}

double \_calculateDistance(double lat1, double lng1, double lat2, double lng2) {

// Haversine Formel für Distanzberechnung in km

const double earthRadius = 6371; // km

final double dLat = \_degreesToRadians(lat2 - lat1);

final double dLng = \_degreesToRadians(lng2 - lng1);

final double a =

sin(dLat / 2) \* sin(dLat / 2) +

cos(\_degreesToRadians(lat1)) \* cos(\_degreesToRadians(lat2)) \*

sin(dLng / 2) \* sin(dLng / 2);

final double c = 2 \* atan2(sqrt(a), sqrt(1 - a));

return earthRadius \* c;

}

double \_degreesToRadians(double degrees) {

return degrees \* (pi / 180);

}

}

// Helper Model für User Activity

class UserActivity {

final String id;

final String activityType;

final String entityId;

final String entityType;

final DateTime timestamp;

UserActivity({

required this.id,

required this.activityType,

required this.entityId,

required this.entityType,

required this.timestamp,

});

Map<String, dynamic> toJson() => {

'id': id,

'activityType': activityType,

'entityId': entityId,

'entityType': entityType,

'timestamp': timestamp.toIso8601String(),

};

factory UserActivity.fromJson(Map<String, dynamic> json) => UserActivity(

id: json['id'],

activityType: json['activityType'],

entityId: json['entityId'],

entityType: json['entityType'],

timestamp: DateTime.parse(json['timestamp']),

);

}

**2. GitHub Actions Deployment**

**2.1 .github/workflows/deploy.yml**

name: Deploy Flutter Web to GitHub Pages

on:

push:

branches: [ main ]

pull\_request:

branches: [ main ]

# Sets permissions of the GITHUB\_TOKEN to allow deployment to GitHub Pages

permissions:

contents: read

pages: write

id-token: write

# Allow only one concurrent deployment, skipping runs queued between the run inprogress

and latest queued

concurrency:

group: "pages"

cancel-in-progress: false

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout Repository

uses: actions/checkout@v4

- name: Setup Flutter

uses: subosito/flutter-action@v2

with:

flutter-version: '3.16.0'

channel: 'stable'

- name: Get Flutter Dependencies

run: flutter pub get

- name: Generate Code (build\_runner)

run: flutter pub run build\_runner build --delete-conflicting-outputs

- name: Generate Assets

run: flutter packages pub run flutter\_gen

- name: Test Flutter App

run: flutter test

- name: Build Flutter Web

run: |

flutter build web --release --web-renderer html --base-href "/flashfeed-prototype/"

- name: Setup Pages

uses: actions/configure-pages@v4

- name: Upload Build Artifacts

uses: actions/upload-pages-artifact@v3

with:

path: 'build/web'

deploy:

environment:

name: github-pages

url: ${{ steps.deployment.outputs.page\_url }}

runs-on: ubuntu-latest

needs: build

if: github.ref == 'refs/heads/main'

steps:

- name: Deploy to GitHub Pages

id: deployment

uses: actions/deploy-pages@v4

**2.2 web/index.html (Flutter Web Konfiguration)**

<!DOCTYPE html>

<html>

<head>

<!--

If you are serving your web app in a path other than the root, change the

href value below to reflect the base path you are serving from.

The path provided below has to start and end with a slash "/" in order for

it to work correctly.

For more details:

\* https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base

This is a placeholder for base href that will be replaced by the value of

the `--base-href` argument provided to `flutter build`.

-->

<base href="$FLUTTER\_BASE\_HREF">

<meta charset="UTF-8">

<meta content="IE=Edge" http-equiv="X-UA-Compatible">

<meta name="description" content="FlashFeed - Sustainable Real-Time Grocery Deals

Platform">

<!-- iOS meta tags & icons -->

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-title" content="FlashFeed">

<link rel="apple-touch-icon" href="icons/Icon-192.png">

<!-- Favicon -->

<link rel="icon" type="image/png" href="favicon.png"/>

<title>FlashFeed Prototype</title>

<link rel="manifest" href="manifest.json">

<!-- PWA Meta Tags -->

<meta name="theme-color" content="#2E8B57">

<meta name="viewport" content="width=device-width, initial-scale=1.0, userscalable=

no">

<!-- Google Maps API -->

<script

src="https://maps.googleapis.com/maps/api/js?key=YOUR\_API\_KEY&libraries=places">

</script>

<!-- Service Worker Registration -->

<script>

if ('serviceWorker' in navigator) {

window.addEventListener('flutter-first-frame', function () {

navigator.serviceWorker.register('flutter\_service\_worker.js');

});

}

</script>

</head>

<body>

<!-- Loading Screen -->

<div id="loading" style="

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: #2E8B57;

display: flex;

justify-content: center;

align-items: center;

z-index: 9999;

">

<div style="text-align: center; color: white;">

<div style="

width: 60px;

height: 60px;

border: 4px solid rgba(255, 255, 255, 0.3);

border-top: 4px solid white;

border-radius: 50%;

animation: spin 1s linear infinite;

margin: 0 auto 20px;

"></div>

<h2 style="margin: 0; font-family: 'Roboto', sans-serif; font-weight:

300;">FlashFeed</h2>

<p style="margin: 10px 0 0; font-family: 'Roboto', sans-serif; opacity: 0.8;">Lädt

Prototyp...</p>

</div>

</div>

<style>

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

body {

margin: 0;

padding: 0;

font-family: 'Roboto', sans-serif;

background-color: #FAFAFA;

overflow-x: hidden;

}

/\* Hide loading screen when Flutter is ready \*/

.flutter-ready #loading {

display: none;

}

</style>

<script src="flutter.js" defer></script>

<script>

window.addEventListener('load', function(ev) {

// Download main.dart.js

\_flutter.loader.loadEntrypoint({

serviceWorker: {

serviceWorkerVersion: serviceWorkerVersion,

},

onEntrypointLoaded: function(engineInitializer) {

engineInitializer.initializeEngine().then(function(appRunner) {

// Hide loading screen

document.body.classList.add('flutter-ready');

appRunner.runApp();

});

}

});

});

</script>

</body>

</html>

**2.3 web/manifest.json (PWA Konfiguration)**

{

"name": "FlashFeed Prototype",

"short\_name": "FlashFeed",

"start\_url": ".",

"display": "standalone",

"background\_color": "#FAFAFA",

"theme\_color": "#2E8B57",

"description": "Sustainable Real-Time Grocery Deals Platform - Academic Prototype",

"orientation": "portrait-primary",

"prefer\_related\_applications": false,

"icons": [

{

"src": "icons/Icon-192.png",

"sizes": "192x192",

"type": "image/png"

},

{

"src": "icons/Icon-512.png",

"sizes": "512x512",

"type": "image/png"

},

{

"src": "icons/Icon-maskable-192.png",

"sizes": "192x192",

"type": "image/png",

"purpose": "maskable"

},

{

"src": "icons/Icon-maskable-512.png",

"sizes": "512x512",

"type": "image/png",

"purpose": "maskable"

}

]

}

**3. Asset Management & Build Scripts**

**3.1 assets/data/mock\_chains.json**

[

{

"id": "edeka",

"name": "EDEKA",

"displayName": "EDEKA",

"logoUrl": "assets/images/logos/edeka.png",

"primaryColor": "#005CA9",

"website": "https://www.edeka.de",

"customerServicePhone": "0800-3335225",

"isActive": true,

"storeCount": 8,

"categoryMappings": {

"dairy": "Molkereiprodukte",

"meat": "Fleisch & Wurst",

"fruits": "Obst & Gemüse",

"bakery": "Bäckerei",

"beverages": "Getränke"

}

},

{

"id": "rewe",

"name": "REWE",

"displayName": "REWE",

"logoUrl": "assets/images/logos/rewe.png",

"primaryColor": "#CC071E",

"website": "https://www.rewe.de",

"customerServicePhone": "0221-1491",

"isActive": true,

"storeCount": 8,

"categoryMappings": {

"dairy": "Milch & Käse",

"meat": "Fleisch & Geflügel",

"fruits": "Frische",

"bakery": "Backshop",

"beverages": "Getränke"

}

},

{

"id": "aldi",

"name": "ALDI SÜD",

"displayName": "ALDI",

"logoUrl": "assets/images/logos/aldi.png",

"primaryColor": "#00549F",

"website": "https://www.aldi-sued.de",

"customerServicePhone": "0800-2534638",

"isActive": true,

"storeCount": 6,

"categoryMappings": {

"dairy": "Milchprodukte",

"meat": "Fleischwaren",

"fruits": "Obst/Gemüse",

"bakery": "Backwaren",

"beverages": "Getränke"

}

},

{

"id": "lidl",

"name": "LIDL",

"displayName": "LIDL",

"logoUrl": "assets/images/logos/lidl.png",

"primaryColor": "#0050AA",

"website": "https://www.lidl.de",

"customerServicePhone": "0800-4353535",

"isActive": true,

"storeCount": 7,

"categoryMappings": {

"dairy": "Milch & Molkereiprodukte",

"meat": "Fleisch & Wurst",

"fruits": "Obst & Gemüse",

"bakery": "Brot & Gebäck",

"beverages": "Getränke"

}

},

{

"id": "netto\_schwarz",

"name": "NETTO",

"displayName": "Netto",

"logoUrl": "assets/images/logos/netto-schwarz.png",

"primaryColor": "#FFD100",

"website": "https://www.netto-online.de",

"customerServicePhone": "0800-2000015",

"isActive": true,

"storeCount": 5,

"categoryMappings": {

"dairy": "Milch & Joghurt",

"meat": "Fleisch & Aufschnitt",

"fruits": "Frisches Obst & Gemüse",

"bakery": "Backwaren",

"beverages": "Getränke"

}

}

]

**3.2 Build Script - scripts/build.sh**

#!/bin/bash

# FlashFeed Build Script

echo "

􂶱􂶲􂶳􂶴􂶵􂶶􂶷􂶸􂶹 Starting FlashFeed Prototype Build..."

# Check if Flutter is installed

if ! command -v flutter &> /dev/null; then

echo "

􁤶􁤷 Flutter is not installed. Please install Flutter first."

exit 1

fi

# Check Flutter doctor

echo "

􀴹􀴺􀴻􀴼􀴽􀴾 Running Flutter Doctor..."

flutter doctor

# Get dependencies

echo "

􀵪􀵫􀵬􀵭􀵮􀵯􀵰􀵱􀵲 Getting Flutter dependencies..."

flutter pub get

# Generate code

echo "

􀱆􀱇􀱈􀱉􀱊􀱋􀱌􀱍􀱎 Generating code with build\_runner..."

flutter pub run build\_runner build --delete-conflicting-outputs

# Generate assets

echo "

􁀺􁀻􁀼􁀽􁀾􁀿􁁀 Generating asset references..."

flutter packages pub run flutter\_gen

# Run tests

echo "

􂦂􂦃􂦄􂦅 Running tests..."

flutter test

# Build for web

echo "

􀯲􀯳􀯴 Building Flutter Web..."

flutter build web --release --web-renderer html --base-href "/flashfeed-prototype/"

# Create deployment info

echo "

􁟨􁟩􁟪􁟫􁟬􁟭􁟮􁟯􁟰 Creating deployment info..."

cat > build/web/deployment\_info.json << EOF

{

"buildDate": "$(date -u +"%Y-%m-%dT%H:%M:%SZ")",

"version": "1.0.0",

"platform": "web",

"environment": "production",

"repository": "flashfeed-prototype"

}

EOF

echo "

􀿨􀿩􀿪 Build completed successfully!"

echo "

􀚙􀚚􀚛 Web build available in: build/web/"

echo "

􀛭􀛮􀛯􀛰 Ready for GitHub Pages deployment"

**3.3 QR Code Generator Script - scripts/generate\_qr.py**

#!/usr/bin/env python3

"""

QR Code Generator für FlashFeed Prototype

Generiert QR Code für GitHub Pages URL

"""

import qrcode

import qrcode.image.svg

from pathlib import Path

def generate\_qr\_code():

# GitHub Pages URL (ersetzen Sie YOUR\_USERNAME)

url = "https://YOUR\_USERNAME.github.io/flashfeed-prototype/"

# QR Code erstellen

qr = qrcode.QRCode(

version=1,

error\_correction=qrcode.constants.ERROR\_CORRECT\_L,

box\_size=10,

border=4,

)

qr.add\_data(url)

qr.make(fit=True)

# PNG Version

img = qr.make\_image(fill\_color="black", back\_color="white")

# Output Verzeichnis erstellen

output\_dir = Path("docs")

output\_dir.mkdir(exist\_ok=True)

# PNG speichern

png\_path = output\_dir / "flashfeed\_qr\_code.png"

img.save(png\_path)

# SVG Version

factory = qrcode.image.svg.SvgPathImage

svg\_img = qr.make\_image(image\_factory=factory)

svg\_path = output\_dir / "flashfeed\_qr\_code.svg"

svg\_img.save(svg\_path)

# HTML Seite mit QR Code

html\_content = f"""

<!DOCTYPE html>

<html>

<head>

<title>FlashFeed Prototype - QR Code</title>

<style>

body {{

font-family: Arial, sans-serif;

text-align: center;

margin: 50px;

background-color: #FAFAFA;

}}

.container {{

max-width: 600px;

margin: 0 auto;

background: white;

padding: 40px;

border-radius: 12px;

box-shadow: 0 4px 12px rgba(0,0,0,0.1);

}}

.qr-code {{

margin: 30px 0;

}}

.url {{

background: #f5f5f5;

padding: 10px;

border-radius: 8px;

font-family: monospace;

word-break: break-all;

margin: 20px 0;

}}

h1 {{

color: #2E8B57;

margin-bottom: 10px;

}}

.subtitle {{

color: #666;

margin-bottom: 30px;

}}

</style>

</head>

<body>

<div class="container">

<h1>

􀹞􀹠􀹡 FlashFeed Prototype</h1>

<p class="subtitle">Scannen Sie den QR Code für sofortigen Zugang</p>

<div class="qr-code">

<img src="flashfeed\_qr\_code.png" alt="QR Code für FlashFeed Prototype"

style="max-width: 300px;">

</div>

<p><strong>URL:</strong></p>

<div class="url">{url}</div>

<p><em>Funktioniert auf allen modernen Smartphones und Tablets</em></p>

<hr style="margin: 40px 0; border: 1px solid #eee;">

<h3>

􁏟􁏠􁏡􁏢􁏣􁏤􁏥 Wie verwenden?</h3>

<ol style="text-align: left; max-width: 400px; margin: 0 auto;">

<li>QR Code mit Smartphone-Kamera scannen</li>

<li>Link in Browser ö􀆯nen</li>

<li>FlashFeed Prototype testen</li>

<li>Alle drei Panels durchnavigieren</li>

<li>Flash-Rabatte und Lageplan-Demo ausprobieren</li>

</ol>

</div>

</body>

</html>

"""

html\_path = output\_dir / "qr\_code.html"

with open(html\_path, 'w', encoding='utf-8') as f:

f.write(html\_content)

print(f"

􀿨􀿩􀿪 QR Code generiert:")

print(f"

􁏟􁏠􁏡􁏢􁏣􁏤􁏥 PNG: {png\_path}")

print(f"

􁀺􁀻􁀼􁀽􁀾􁀿􁁀 SVG: {svg\_path}")

print(f"

􀯲􀯳􀯴 HTML: {html\_path}")

print(f"

􀠻􀠼􀠽 URL: {url}")

if \_\_name\_\_ == "\_\_main\_\_":

generate\_qr\_code()

**4. Documentation**

**4.1 README.md**

#

􀹞􀹠􀹡 FlashFeed Prototype

Eine nachhaltige Echtzeit-Rabatt-Plattform für Lebensmittel als Flutter Web-Prototyp für

eine wissenschaftliche Arbeit.

##

􂶱􂶲􂶳􂶴􂶵􂶶􂶷􂶸􂶹 Live Demo

\*\*[FlashFeed Prototype](https://YOUR\_USERNAME.github.io/flashfeed-prototype/)\*\*

Scannen Sie den QR Code für sofortigen Zugang auf dem Smartphone:

![QR Code](docs/flashfeed\_qr\_code.png)

##

􁏟􁏠􁏡􁏢􁏣􁏤􁏥 Features

### Drei-Panel-Navigation

- \*\*Panel 1: Angebote\*\* - Multi-Händler-Angebotsvergleich mit Produktkategorien

- \*\*Panel 2: Karte\*\* - Interaktive Karte mit Filialstandorten und Radius-Filter

- \*\*Panel 3: Flash-Rabatte\*\* - Echtzeit-Rabatte mit Countdown und Indoor-Navigation

### Kernfunktionalitäten

-

􀲬􀲭􀲮􀲯􀲰􀲱􀲲􀲳 5 simulierte Handelsketten (EDEKA, REWE, ALDI, LIDL, Netto)

-

􀝠􀝡􀝢 40+ Mock-Filialen mit GPS-Koordinaten

-

􀹞􀹠􀹡 15-20 aktive Flash-Rabatte mit automatischer Rotation

-

􁦋􁦌􁦍􁦎􁦏􁦐􁦑􁦒 Indoor-Navigation mit Beacon-Simulation

-

􁏟􁏠􁏡􁏢􁏣􁏤􁏥 Responsive Design für alle Geräte

-

􁍮􁍯 Echtzeit-Updates ohne Server

##

􁢇􁢈􁢉􁢊 Technische Implementierung

### Architektur

- \*\*Frontend:\*\* Flutter Web 3.16+

- \*\*State Management:\*\* BLoC Pattern

- \*\*Storage:\*\* Browser LocalStorage

- \*\*Maps:\*\* Google Maps JavaScript API

- \*\*Deployment:\*\* GitHub Pages + Actions

### Projektstruktur

lib/ ├── core/ # Konstanten, Services, Utils ├── data/ # Models, Repositories, Mock-

Data └── presentation/ # BLoCs, Screens, Widgets

##

􂶱􂶲􂶳􂶴􂶵􂶶􂶷􂶸􂶹 Setup & Installation

### Voraussetzungen

- Flutter 3.16.0 oder höher

- Dart SDK 3.1.0+

- Git

### Installation

```bash

# Repository klonen

git clone https://github.com/YOUR\_USERNAME/flashfeed-prototype.git

cd flashfeed-prototype

# Dependencies installieren

flutter pub get

# Code generieren

flutter pub run build\_runner build --delete-conflicting-outputs

# Assets generieren

flutter packages pub run flutter\_gen

# Tests ausführen

flutter test

# Development Server starten

flutter run -d chrome

**Build für Deployment**

# Web Build

flutter build web --release --web-renderer html --base-href "/flashfeed-prototype/"

# Mit Build-Script

chmod +x scripts/build.sh

./scripts/build.sh

**􀝓􀝔􀝕􀝖􀝗􀝘 Mock-Daten-System**

Das Prototyp-System generiert und verwaltet realistische Mock-Daten:

 **500+ Produkte** über 5 Kategorien

 **Timer-basierte Updates** alle 2 Stunden für Flash-Rabatte

 **Realistische Preise** und Rabattstrukturen (30-70%)

 **Beacon-Simulation** für Indoor-Navigation

 **User-Activity-Tracking** für Demo-Analytics

**􁇤􁇥􁇦􁇧􁇨 Verwendung**

**Für Professor-Demonstration**

1. QR Code scannen oder URL ö􀆯nen

2. App lädt automatisch Mock-Daten

3. Drei Panels durchnavigieren:

o Angebote: Händler auswählen, Kategorien durchsuchen

o Karte: Filialen in verschiedenen Städten erkunden

o Flash: Countdown-Timer und "Filiale betreten"-Button testen

**Für Entwickler**

// Neue Mock-Daten generieren

final mockService = MockDataService();

await mockService.initializeMockData(storageService);

// Flash-Deals manuell aktualisieren

context.read<FlashDealsBloc>().add(RefreshFlashDealsEvent());

**􀵚􀵛􀵜􀵝􀵞􀵠􀵡􀵢􀵣 Wissenschaftlicher Kontext**

Dieser Prototyp demonstriert die Konzepte aus der Business Model Canvas-Analyse

einer nachhaltigen Lebensmittel-Echtzeitangebots-Plattform für eine wissenschaftliche

Arbeit.

**Validierte Geschäftsmodell-Komponenten**

 Two-sided Market (B2B/B2C)

 Freemium-Monetarisierung

 Netzwerke􀆯ekte durch Multi-Händler-Integration

 Food Waste Reduction durch Echtzeit-Rabatte

 Location-based Services mit Indoor-Navigation

**􂶱􂶲􂶳􂶴􂶵􂶶􂶷􂶸􂶹 Deployment**

Das Projekt ist für automatisches Deployment auf GitHub Pages konfiguriert:

1. Repository zu GitHub pushen

2. GitHub Pages in Settings aktivieren

3. GitHub Actions übernimmt automatisches Deployment

4. QR Code für mobile Demo nutzen

**􁔎􁔏􁔐 Support**

Bei Fragen zur Implementierung oder zum wissenschaftlichen Konzept:



􁟾􁟿􁠀􁠁􁠂 E-Mail: [Ihre Kontaktdaten]



􀵚􀵛􀵜􀵝􀵞􀵠􀵡􀵢􀵣 Dokumentation: [Link zu vollständigen Requirements]

**Version:** 1.0.0

**Zweck:** Wissenschaftlicher Prototyp

**Technologie:** Flutter Web + GitHub Pages

### 4.2 docs/DEPLOYMENT.md

```markdown

# FlashFeed Deployment Anleitung

## GitHub Pages Deployment

### 1. Repository Setup

```bash

# Repository erstellen auf GitHub

# Repository klonen

git clone https://github.com/YOUR\_USERNAME/flashfeed-prototype.git

cd flashfeed-prototype

# Projektdateien hinzufügen

git add .

git commit -m "Initial FlashFeed prototype"

git push origin main

**2. GitHub Pages aktivieren**

1. GitHub Repository ö􀆯nen

2. Settings → Pages

3. Source: "GitHub Actions" auswählen

4. Workflow wird automatisch erkannt

**3. Build-Konfiguration anpassen**

In .github/workflows/deploy.yml:

# Base href für Ihr Repository anpassen

flutter build web --release --web-renderer html --base-href "/IHR\_REPOSITORY\_NAME/"

**4. Google Maps API Key**

1. Google Cloud Console → APIs & Services

2. Maps JavaScript API aktivieren

3. API Key erstellen

4. In web/index.html einsetzen:

<script

src="https://maps.googleapis.com/maps/api/js?key=IHR\_API\_KEY&libraries=places"></

script>

**5. QR Code generieren**

# Python-Abhängigkeit installieren

pip install qrcode[pil]

# QR Code generieren

python3 scripts/generate\_qr.py

**6. Deployment überwachen**

 GitHub Actions Tab für Build-Status

 Pages-URL: https://YOUR\_USERNAME.github.io/flashfeed-prototype/

**Lokale Entwicklung**

**Development Server**

flutter run -d chrome --web-port 3000

**Hot Reload aktiviert**

flutter run -d chrome --hot

**Build Testen**

# Lokaler Build

flutter build web --web-renderer html

# Lokaler Server für Testing

cd build/web

python3 -m http.server 8000

URL: http://localhost:8000

**Troubleshooting**

**Build Fails**

# Cache löschen

flutter clean

flutter pub get

# Dependencies neu installieren

rm pubspec.lock

flutter pub get

flutter pub run build\_runner build --delete-conflicting-outputs

**GitHub Pages nicht erreichbar**

 Repository Settings → Pages überprüfen

 GitHub Actions Logs prüfen

 DNS-Propagation abwarten (bis zu 24h)

**Mobile Performance**

 Web-Renderer auf "html" statt "canvaskit" setzen

 Service Worker für O􀆯line-Funktionalität

 Lazy Loading für große Assets implementieren